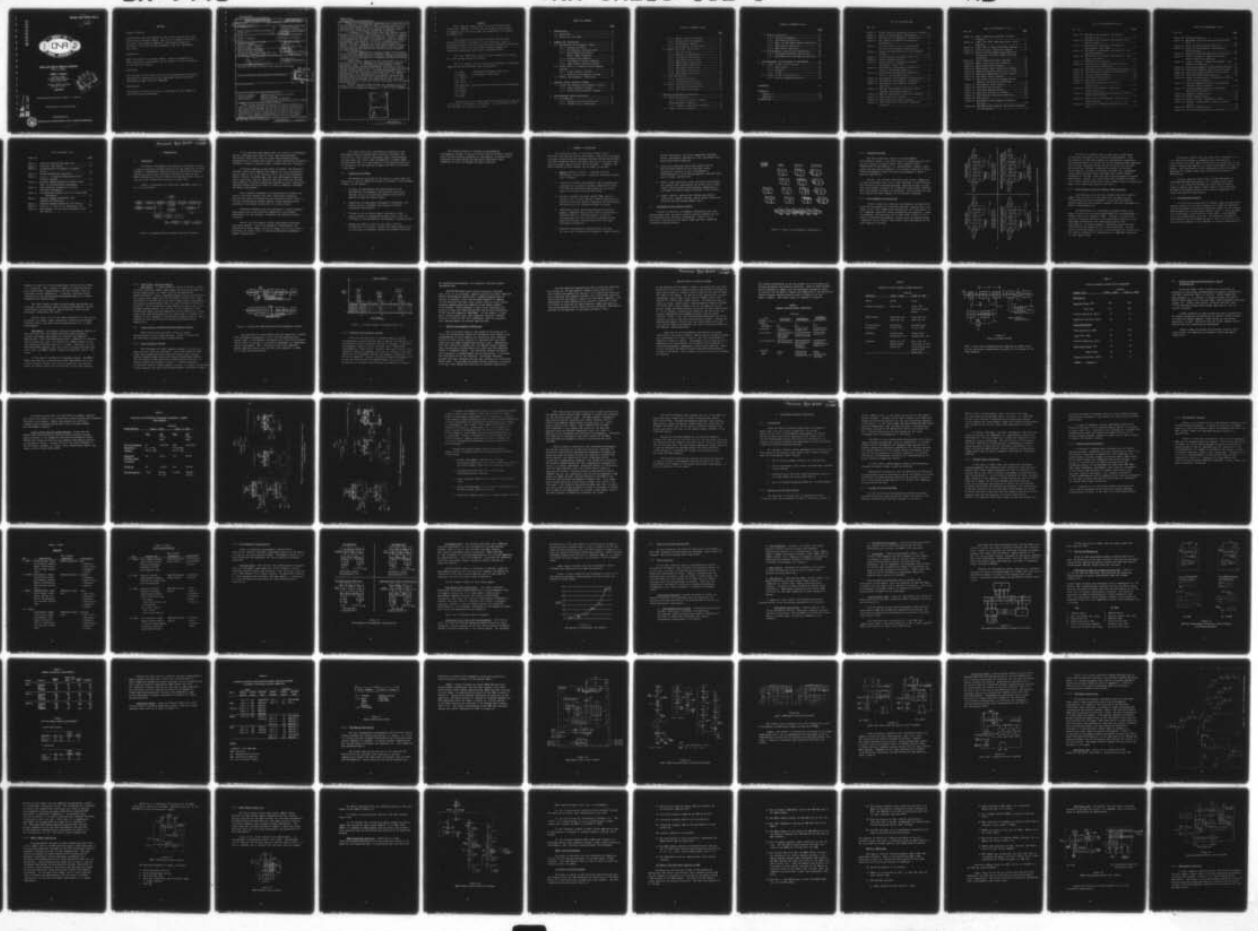
AD-A042 466 RAYTHEON CO BEDFORD MASS MISSILE SYSTEMS DIV F/G 16/4.1
MODULAR DIGITAL MISSILE GUIDANCE PHASE III.(U)
MAY 77 F J LANGLEY N00014-75-C-0549
UNCLASSIFIED BR-9448 ONR-CR233-052-3 NL

1 OF 3
AD
A042466



REPORT ONR-CR233-052-3

12 B.S.

MODULAR DIGITAL MISSILE GUIDANCE

PHASE III REPORT

FRANK J. LANGLEY

Raytheon Company
Missile Systems Division
Bedford Mass 01730

Contract N00014-75-C-0549
ONR Task 233-052

4 MAY 1977

Technical Report for Period 20 NOV 75 - 31 Dec 76

PREPARED FOR THE

OFFICE OF NAVAL RESEARCH ● 800 N. QUINCY ST. ● ARLINGTON ● VA ● 22217

see 1473

NOTICES

Change of Address

Organizations receiving reports on the initial distribution list should confirm correct address. This list is located at the end of the report. Any change of address or distribution should be conveyed to the Office of Naval Research, Code 212, Washington, D.C. 22217.

Disposition

When this report is no longer needed, it may be transmitted to other authorized organizations. Do not return it to the originator or the monitoring office.

Disclaimer

The findings in this report are not to be construed as an official Department of Defense or Military Department position unless so designated by other official documents.

Reproduction

Reproduction in whole or in part is permitted for any purpose of the United States Government.

(9) Technical rept. 20 Nov 75 - 31 Dec 76,

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (18) ONR-CR233-052-3 (19)	2. GOVT ACCESSION NO. -
	3. RECIPIENT'S CATALOG NUMBER -
4. TITLE (and Subtitle) (6) Modular Digital Missile Guidance. Phase III.	5. TYPE OF REPORT & PERIOD COVERED Technical Report 20 Nov 75 to 31 Dec 76
	6. PERFORMING ORG. REPORT NUMBER (14) BR-9448
7. AUTHOR(s) (10) Frank J. Langley et al	8. CONTRACT OR GRANT NUMBER(s) (15) N00014-75-C-0549
9. PERFORMING ORGANIZATION NAME AND ADDRESS Raytheon Company Missile Systems Division Bedford, Mass 01730	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62332N RF32-311-801, 3-29 2311-801
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Vehicle Warfare Technology Code 212 Arlington, Virginia 22217	12. REPORT DATE (11) 4 May 1977
	13. NUMBER OF PAGES 264 (12) 262 p.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (16) F32311 (17) RF32311801	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

DDC AUG 5 1977 C

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Missile Guidance
Digital Control
Microcomputer
Federated Microcomputer Systems
Modular Systems
Standard Electronic Modules
Very Large Scale Integrated Circuits (VLSI)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report presents the results of the third phase of a study to investigate the feasibility of modular digital guidance and control systems for air-to-air missile applications. The studies involved the analysis of functions for digital implementation in all classes of air-to-air missiles and the derivation of computer requirements in terms of throughput memory, architectural features, modularity and compatible software characteristics.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

297620

Phase III validated the performance and effectiveness of the macromodular microcomputer family defined in Phase II, on an individual module basis; as whole microcomputers; and as federated microcomputer systems applied to specific generic missile types, using digital simulation techniques.

In summary, the studies have shown that modular digital guidance and control is both feasible and effective in improving missile performance and flexibility to counteract changing threat situations and advancing technology. Using a common microcomputer bus interface, (microbus), a family of fourteen microcomputer macromodules, in various configurations, will support the entire range of air-to-air missile functions. Further, the Navy standard electronic module (SEM), in either SEM-1A or SEM-2A configurations, provides a practical means of packaging the macromodules and maintaining the standard microbus interface.

Federated microcomputer systems provide a better match of major missile functions with computer capability, providing the desired subsystem autonomy for modular design, manufacture, assembly, test, maintenance and subsequent modification without system disruption.

Target seeker signal processing requirements for air-to-air missiles are low, compared to avionics and air defense processing loads. In two federated microcomputer systems analyzed, a common 16-bit microcomputer was found to satisfy the performance requirements for target seeker signal processing, estimation and guidance, (i.e. the steering command loop), by adding a hardware fast Fourier transform (μFFT) macromodule to the microbus. The remaining missile functions, i.e. seeker gimballed platform stabilization and control, autopilot and fuzing/telemetry could each be satisfied with the 8-bit byte microprocessor macromodule (MIL 8080-based) with a microbus multiplier module to meet the required throughput for the higher speed functions.

Support software should be that of a mature 16-bit mini computer, (e.g. AN/AYK-14, PDP-11/34), and the 8080 package, as should the programming language, i.e. CMS-2, PL/M, FORTRAN IV.

PREFACE

This technical report covers the work performed under Contract No. N00014-75-C-0549 from 20 November 1975 through 31 December 1976. The first draft was submitted by the authors on 30 May 1977.

The purpose of this contract together with the work performed in Phase I and Phase II was to provide a basis for designing modular air-to-air missile guidance systems with improved performance, flexibility and growth features compared to traditional analog and early digital implementations.

Cdr. P.R. "Bob" Hite, Office of Naval Research, Arlington, Va., was the Navy Scientific Officer.

Mr. F.J. Langley was the Principal Investigator for Raytheon supported by the following study team members:

P.C. Barr	2901-Based μFFT Module Definition
J. Sheehan	Microcomputer Simulations
A.J. Mannion	Microprocessor Architectures vs Performance
R.G. Carle	
E.O. Morrow	
M.B. Robertson	
P. Hilcoff	
J.P. Newell	
K.D. Laube	
B. Dodson	

Publication of this report does not constitute Navy approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont)

LIST OF ILLUSTRATIONS (Cont)

LIST OF ILLUSTRATIONS (Cont)

LIST OF ILLUSTRATIONS (Cont)

LIST OF TABLES

LIST OF TABLES (Cont)

1. INTRODUCTION

1.1 Background

The design, development and production of missiles to cover a range of presently defined missions with the capability of being upgraded to accommodate changing threat situations and advancing technology without major redesign, stresses the need for more modular guidance and control electronics possessing both physical and electrical flexibility features at lowest cost.

Figure 1 illustrates the functional complement typical of air to air missiles.

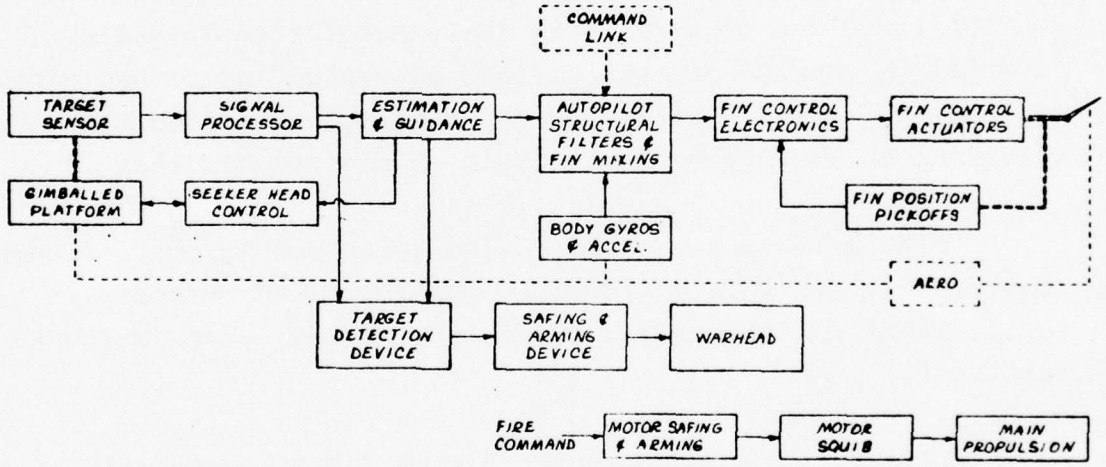


Figure 1. On-Board Missile Guidance and Control System

In the previous study phases (Ref. R-1 and R-2), programmable digital techniques were shown to offer improved performance and greater flexibility than the traditional hardwired analog implementations of seeker head control, signal processing, estimation, guidance, autopilot, warhead fuzing, telemetry and test functions.

To achieve modularity and growth in hardware and software, a top-down system study approach was adopted, by first dividing the entire range of air to air missiles into a set of distinguishable generic classes, including upper and lower performance boundaries within each class, then by: defining the major functions and data rates amenable to digital processing, determining their constituent software modules and sizing these in terms of computer throughput and memory requirements.

Such a modular breakdown of on-board missile guidance and control functions together with their associated interfaces, provided the option of configuring and evaluating either single or multiple federated/distributed computer system implementations according to the design constraints of a given missile.

Simulation analyses were also performed to confirm computer requirements and relate algorithm complexity to performance improvements for the guidance, estimation and autopilot/control functions.

Lastly, with the computer design requirements determined from these studies, a set of microcomputer "macromodules" was defined which would support the entire range of air-to-air missile functions, in either single or multiple/federated microcomputer system configurations.

The study results were subsequently presented at two government/industry microprocessor standards workshops (Ref. R-3 and R-4), and at several government agency establishments including Navy, Air Force, Army and NASA/JPL. A summary paper was published (Ref. R-5) and presented at the American Institute of Aeronautics and Astronautics Guidance and Control Conference in San Diego.

1.2 Objectives and Scope

The objectives and scope of the Phase III study under the extension of contract N00014-75-C-0549, as defined in the Statement of Work, are as follows:

1. Validate the performance and effectiveness of the macromodular microcomputer configurations previously defined, both on an individual computer basis and as applied to specific missile types.

2. Demonstrate through simple macromodule replacement, the modular growth of hardware and software to obtain performance improvements.

3. Through an initial design phase, configure a Class I missile system for optimum performance using a best fit technique of the macromodular microcomputer configurations.

4. Demonstrate modular growth of the Class I missile configuration to a Class II missile by simple substitution of alternate microcomputer modules and software functions.

The intention being to validate the macromodular microcomputer concept to the point where realistic product function specifications could be prepared for each module, to control their subsequent detailed design and fabrication and ensure their compatibility with the standard microbus interface.

2. SUMMARY & CONCLUSIONS

As a follow-on effort to the initial Phase I and II functional analyses and modular microcomputer definition, the Phase III study has performed the necessary top-down and bottom-up system application-computer module validation exercise to the point where realistic module specifications can be prepared. In summary, the major findings of these studies can be stated as follows:

1. Modular digital is better -- provides improved performance, flexibility, reliability and growth without major re-design.

2. A family of fourteen microcomputer VLSI "macromodules", integrated by a standard microbus, support the system interface and processing requirements of the entire range of air-to-air missiles identified.

3. The Navy standard electronic module (SEM), plug-in circuit board in either SEM-1A or SEM-2A configuration, provides a practical means of packaging the macromodules and maintaining the standard microbus interface.

4. Complimentary metal-oxide semiconductor/silicon-on-sapphire (CMOS/SOS) MSI and LSI circuit equivalents of standard-industry, Schottky-bipolar, bit-slice microprocessor circuits, provide the necessary high-speed and low-power characteristics for the higher performance maromodules.

5. Federated microcomputer systems obviate the need for very high throughput microcomputers, support/enforce

software modularity, and allow independent subsystem design, development, manufacture, test, maintenance and future modification/updating.

6. A practical federated system which meets both the desired system modularity and interface timing constraints, consists of four microcomputers interconnected via an avionics-compatible, MIL-STD-1553A serial digital multiplex bus.

7. Seeker signal processing, state estimation and guidance law computational requirements can be met with a single 16-bit, general-purpose microprocessor supported by a high-speed arithmetic module connected to the microbus.

8. Simple, modular, applications software and proven, widely-used, support software reduce design, coding verification and maintenance/update costs.

2.1 Macromodular Microcomputer Family

A total of fourteen microcomputer macromodules have been defined which cover the entire range of missile guidance and control processing and interface requirements. Figure 2 is a pictorial illustration of the module types and their respective interfacing characteristics.

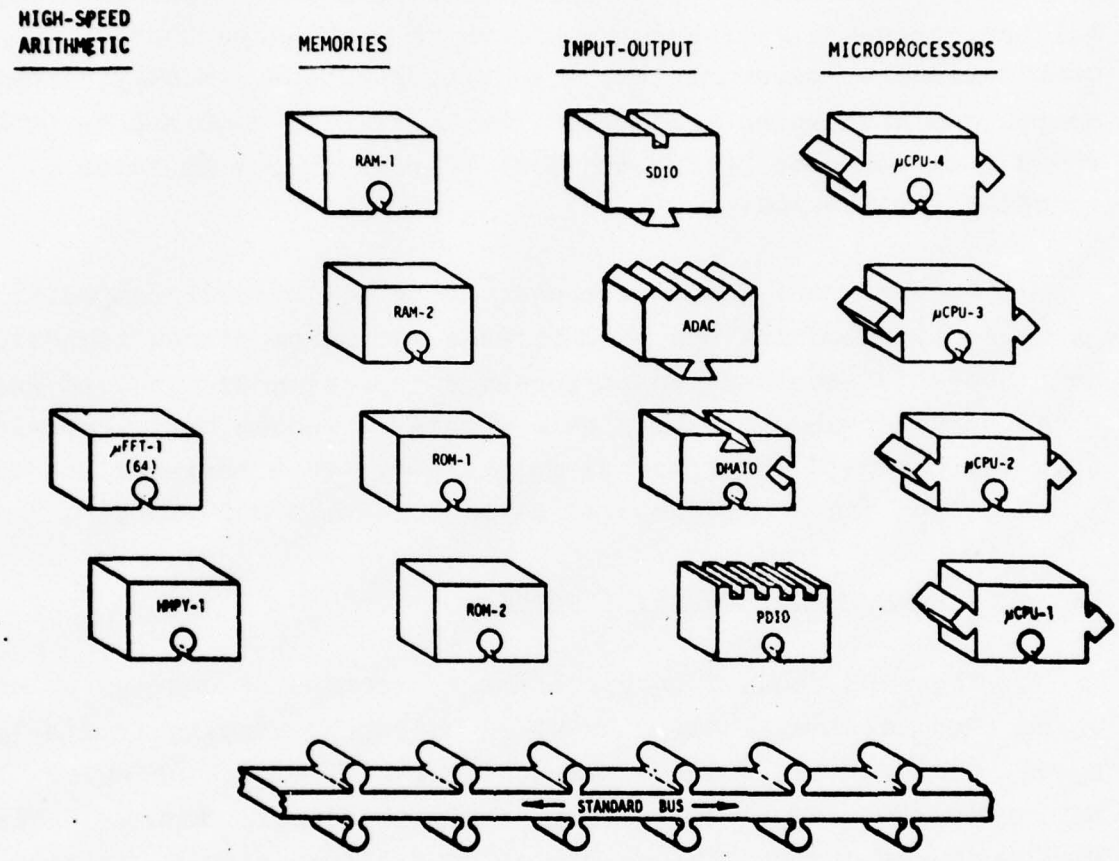


Figure 2 Family of Microcomputer Macromodules

2.1.1 Standard Microbus

The most significant factor in the macromodular microcomputer concept is the definition of a common parallel digital bus, termed a microbus (μBus), which enables various combinations of macromodules, i.e. microprocessor, memory, input-output and high-speed arithmetic, to be readily combined to form a whole microcomputer having the desired performance to match a specific application.

Further, the internal architecture and circuit composition of these macromodules can vary to take advantage of new technology and innovative cost-saving/performance improving designs, as and when these become available, on a module by module basis, provided that the standard interface is maintained. This obviates the need to re-design the whole computer as is currently customary.

2.1.2 Microcomputer Configurations

Figure 3 shows four different microcomputer configurations using the macromodule building-block approach, ranging from a low-speed, 8-bit, Intel 8080A-based configuration to a high-speed, 16-bit, CMOS/SOS, AN/UYK-20 software-compatible microcomputer. These microcomputer configurations can be used either singly, or as a group in federated microcomputer systems using the MIL-STD-1553A-compatible serial digital input-output (SDIO) macromodule as the inter-computer interface.

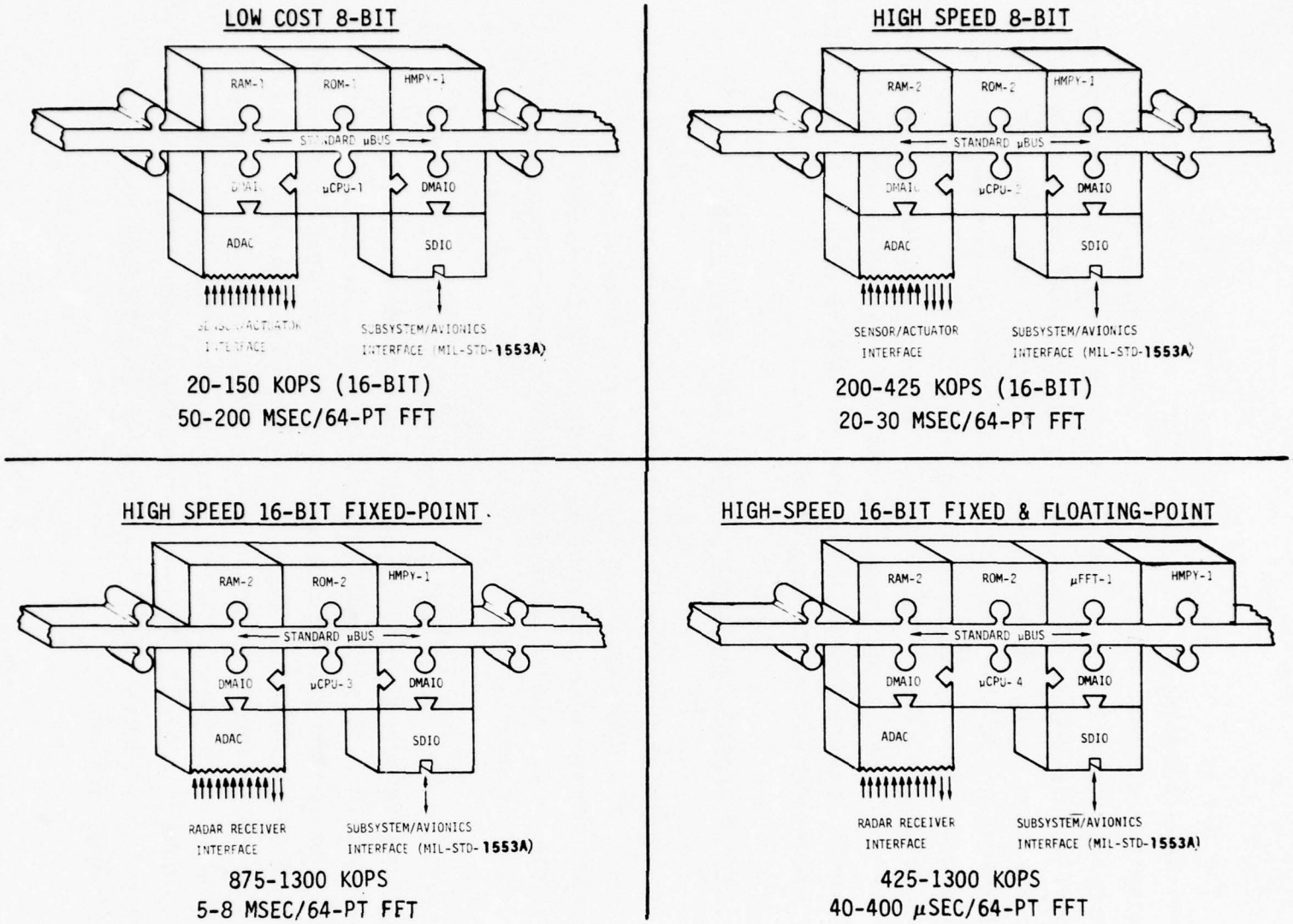


Figure 3 Macromodular Microcomputer Configurations

The use of a standard μBus is based upon a high degree of functional autonomy in the interfacing macromodules. With the exception of RAMs and (P)ROMs, all macromodules incorporate microprogram control to perform the various operating mode sequences and interface responses without action on the part of the μCPU and operational software, excepting the simple initializing of the modules. Further, to minimize overhead (e.g. LOAD/STORE) instructions, optimise useful throughput and reduce frequent access/use of the μBus, each μCPU uses a multiple/general-register architecture. Lastly, all input-output data transfers are made via a direct- memory-access input-output (DMAIO) module incorporating memory block addressing registers and logic.

2.1.3 Navy Standard Electronic Module (SEM) Packaging

A preliminary review of Navy standard electronic modules (SEMs) (Ref. R-6 and R-7), has shown that the SEM-1A and 2A configurations can each accommodate a whole μBus macromodule, thereby supporting the necessary standard interface requirement for module interchangeability.

The SEM modules will accommodate standard-industry LSI memory circuits and the medium-speed N-MOS 8-bit microprocessor in regular dual-in-line packages (DIPs). However, to achieve the necessary low-power dissipation and high packing density for the high-speed macromodules, the complimentary metal oxide semiconductor, silicon-on-sapphire (CMOS/SOS) process has been identified as the compatible LSI circuit technology. Further, existing Schottky-bipolar bit-slice microprocessor LSI circuits are the logical candidates for implementation in CMOS/SOS technology for VLSI applications.

Preliminary studies have also shown that a standard 2 in. x 1 in. hybrid-LSI package is compatible with macrofunction partitioning, e.g. μCPU, RALU, and provides the flexibility to use either a double-sided SEM-1A or a single-sided SEM-2A for each μBus module. Such high-density circuit packaging provides the added advantage of a simple single-layer printed-circuit board for SEM mounting.

Figure 4 shows the three levels of VLSI packaging for high-speed CMOS/SOS macromodules, viz: LSI/MSI circuit chip, 2 in. x 1 in. hybrid-LSI package and either SEM-1A (double-sided) or SEM-2A μBus module, a whole μCPU in the illustration. The SEM-1A macromodule provides the necessary compactness for highly form-factored applications.

2.1.4 Microprocessor Modules

To avoid the proliferation of computer types and associated support software the de-facto standard Intel 8080A microprocessor has been selected as the kernel processing element for the low-end medium-speed 8-bit microprocessor module (μCPU-1). The high-speed 8-bit counterpart is an emulation of the 8080A, using CMOS/SOS bit-slice microprocessor circuits. The Signetics and Advanced Micro Devices bit-slice 8080A emulators, (Refs. R-8 and R-9), are good working examples of this approach to software standardization.

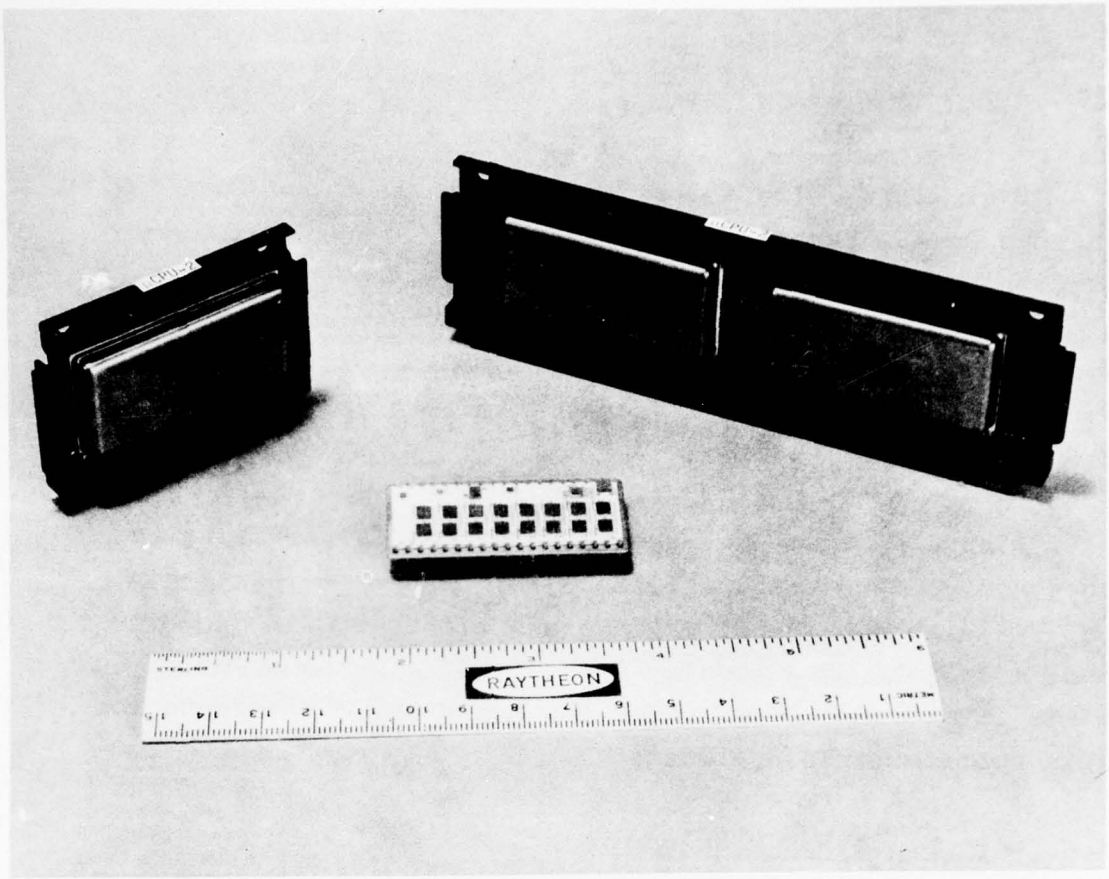


Figure 4 Navy SEM-1A and SEM-2A Microprocessor Macromodules

For the 16-bit microprocessors there is the option of a VLSI CMOS/SOS version of the Navy AN/AYK-14 with the instruction set emulated to the extent necesesary for missile guidance and control applications. Operational software for missile microcomputers would be generated via host compilers/assemblers and would be upward compatible with AN/UYK-20(V) and AN/AYK-14 software.

This approach would support the use of a common programming language and support software for Navy ship, aircraft, and the new generation of missile computers.

2.1.5 High-Speed Arithmetic Modules

FFT Module - Signal processing requirements for radar target seeker air-to-air missiles are low compared to ground-based air defense and avionics system requirements (Figure 5).

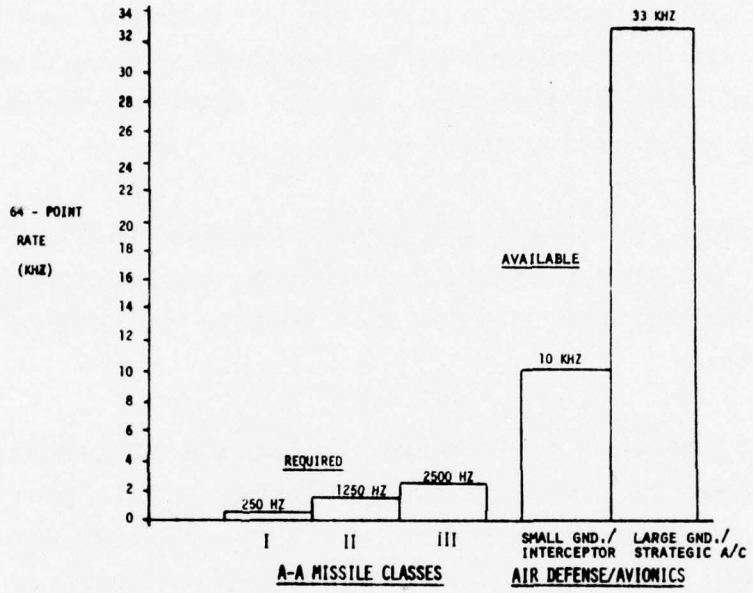


Figure 5 Air-to-Air Missile vs Air Defense and Avionics FFT Throughput Requirements

To avoid the use of an additional front-end FFT processor a μBus compatible FFT processor macromodule has been defined, (μFFT-1), incorporating high-speed read/write, random-access memory (RAM) high-speed register arithmetic and logic (RALU), hardware multiplier (HMPY), and microprogram control unit (μPCU). The RALU

employs the same bit-slice microprocessor circuits as the μCPU-2, -3 & -4 modules. Such a module is capable of executing the 64-point complex FFT in approximately 300 μsec through a "parity-organized" process (Appendix B). A possible lower parts count alternative to the microprocessor-based μFFT module is a charge-coupled device implementation.

The μFFT-1 module bridges the performance gap between the relatively low performance general-purpose microprocessor and the conventional high-performance, special-purpose, front-end FFT processors, eliminating the latter.

The net result of this μBus module definition is the ability to execute all seeker signal processing, estimation and guidance functions within, and under direct program control of, a single microcomputer.

HMPY Module - Performance evaluation programming (Section 6) has shown that the Bus hardware multiply module provides a distinct throughput improvement for multiply time dependent applications. Using the 8080A-based μCPU-1, compared to a software routine taking 126 μsecs, the addition of a HMPY-1 module to the μBus reduces the multiply time to 17.1 μsecs. Similarly, for the high-speed 8080 emulator (Ref. R-8), with a firmware multiply time of 36 μsecs, the HMPY-1 module reduces the execution time to 4.65 μsecs.

In the case of the PDP-11/34 processor however, the HMPY-1 module provides only a 35% multiply speed improvement due to the shorter firmware multiply time and the relatively slow register-memory and memory-memory move instructions of the latter machine.

2.1.6 Input-Output Interface Modules

ADAC Module - A single composite analog to digital, digital to analog conversion (ADAC) module meets all missile guidance and control applications i.e. radar receiver and body motion sensors/actuators. Using a standard-industry form of A-D convertor with short-cycle capability, the high-speed (3 μ sec) 8-bit radar requirement is met and the 10 and 12-bit quantization requirements for gyros and accelerometers over the range of missile classes is satisfied. The D-A conversion requirements can also be met with a standard-industry device i.e. 12-bits, 5 μ sec. conversion time. Simultaneous sample and hold amplifiers are provided on all analog inputs with time multiplexing for A-D conversion in 8 channel increments up to 24 channels. D-A convertor outputs total 8. Spare channels on both the A-D and D-A provide end-around testing capabilities.

2.2 Single Central vs Federated Microcomputer Systems

These studies have provided a choice of system implementation for the system designer. Figure 6 illustrates the two forms of digital missile system design.

2.2.1 Single Computer Systems

The throughput of single computer systems is driven by a short duration "critical time slot". This occurs when the radar target data becomes available toward the end of each 100 msec steering command update period and requires processing together with the regular 2 msec body motion stabilization functions i.e. seeker platform and autopilot, Figure 7. Time-multiplexing of functions within a single computer results in a complex operational software package with many inter-program module interfaces/linkages.

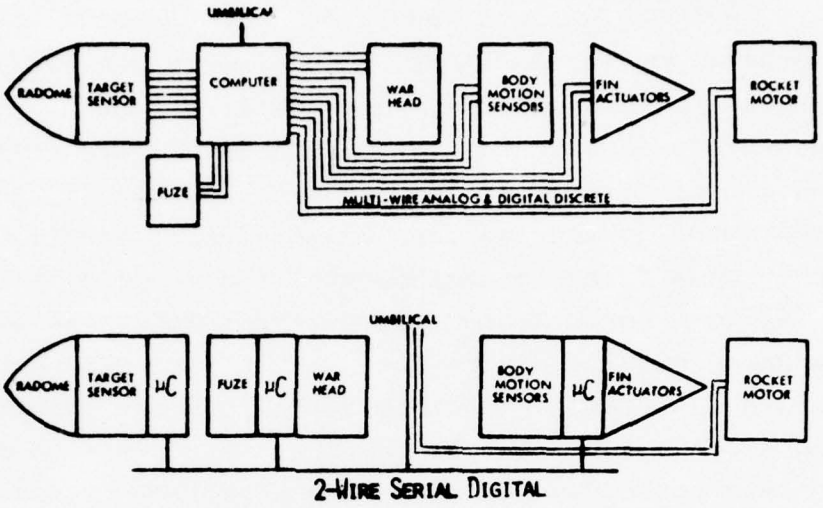


Figure 6 Single and Federated Missile Microcomputer Systems

The questions of large program size and complexity, together with throughput requirements in the 3 million operations per second range, (to accommodate estimating errors and future growth), and the lack of autonomy in the missile subsystems for independent design, development, test, modification and future updating, become important factors when deciding which approach to adopt for a given system application.

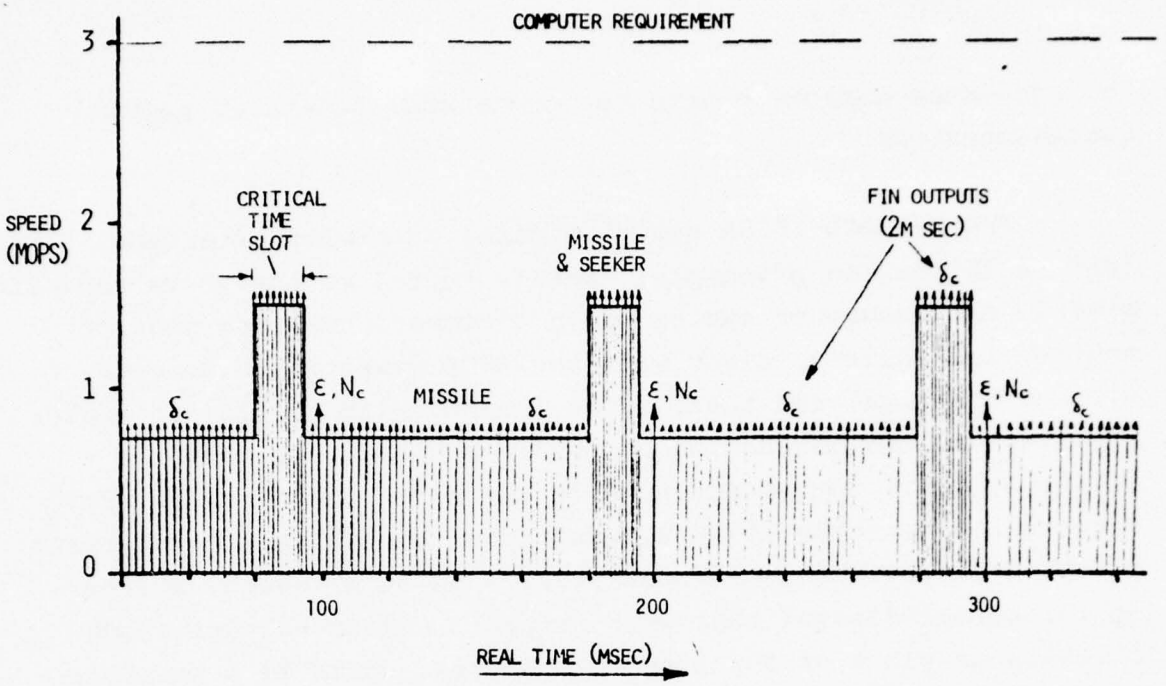


Figure 7. Single Computer Throughput Time Line

2.2.2 Federated Microcomputer Systems

With the declining cost of standard-industry LSI/MSI microprocessor circuits and the need to improve the modularity of new generation missile guidance and control systems, a federated microcomputer system of three, low-cost, 8-bit (8080-based) microcomputers, (head control, autopilot and fuze), and one 16-bit (AN/AYK-14 based) microcomputer provides the necessary subsystem modularity for independent design, development, manufacture, test and future updating. Further, interface between microcomputers is at the low 10/20 Hz steering command update rate, thereby minimizing I/O transfers and achieving the necessary autonomy of

the subsystem microcomputers for successful federated system implementations.

The MIL-STD-1553A serial digital multiplex interface (Ref. R-10) is, in principle, ideally suited to the above federated missile microcomputer system where a command/response mode of control is required, first with the AWCS computer in command via the umbilical and then, after launch, with the 16-bit seeker steering command generation computer as the master computer in the system. Future advances in LSI circuit technology are required to make the 1553A RTU and BCIU the military counterpart of the low-cost commercial UART/USRT. In this regard, a fiber-optic, serial-digital bus, with compatible terminal interface circuits in place of the line transformer, would be a possible cost reducing measure for missile applications.

2.3 Modular Microcomputer Simulations

One of the major tasks in this phase of the study was to verify the architectures chosen for the modules with regard to the reasonableness of their capacity for performing their necessary function as well as their ability to interface with each other using only the common standardized microbus lines. Implementing the state transition diagrams with the Computer Design Language (CDL) simulations clarifies much of the logic and time sequences needed in a modules' design. The simulation becomes the design tool from which a hardware module can be constructed. It provides the description of all the registers, flip-flops, decoders, switches and logic terminals and how they interface at a functional design level. It provides the specifications necessary to continue on to gate level design and from there to hardware construction.

The CDL simulation approach provides an excellent mechanism for testing out the macromodular computer design feasibility. First coding each module and testing it in a stand-alone state and then connecting these modules together into a functional operating entity verifies the logical integrity of the modular approach. Any combination of these modules can be logically connected to study the effects of high and low speed memories, simple and sophisticated microprocessor control logics and the throughput improvements possible by the addition of high-speed arithmetic units.

3. MODULAR DIGITAL GUIDANCE SYSTEMS

At the beginning of the Phase I study it was found that the broad spectrum of air-to-air missiles could be conveniently divided into three major groups or classes, chiefly based on the type of seeker and range of the weapon. At one end of the spectrum are the small lightweight, short-range, single mission weapons using either passive IR or SA-CW radar homing seekers. Mid-spectrum types include the larger, medium-range, high-performance missiles equipped with SA-PD radar target seekers, and, at the upper end of the missile range, multimode missiles constitute the most sophisticated air to air weapons, incorporating semiactive and active PD radar seekers supplemented with command guidance from the launch aircraft. Table 1 lists the distinguishing parameters and characteristics of each of the three major missile classes. Minimum and maximum performance versions were then established within each class and the supporting guidance and control functions identified and computer loads established.

Using the set of microcomputer macromodules previously identified in the Phase II Report, modular digital guidance systems were configured for a Class I missile, and then reconfigured to represent a Class II missile. The purpose of this exercise was to demonstrate the effectiveness of the modular design approach in achieving performance improvements without major redesign. Table 2 lists the major G&C functions and associated program modules, defined in the Phase I and II studies, pertinent to the two classes of missiles.

The greatest differences in the two systems lie in the signal processing estimation and autopilot functions. Both of these systems must be designed to comply with the performance parameters given in Section 3 of the Phase II Report and exemplified in the equivalent digital guidance system structure shown in Figure 12 of the Phase II Report. The latter figure is repeated in Figure 8.

TABLE 1

GENERIC MISSILE FAMILY DEFINITION

FAMILY / PARAMETER	MISSILE CLASS I LOW COST SPECIFIC MISSION DESIGN	MISSILE CLASS II HIGH PERFORMANCE LIMITED MODE DESIGN	MISSILE CLASS III HIGH PERFORMANCE MULTI-MISSION MULTI-MODE DESIGN
ENVELOPE			
RANGE (nmi)	10	30	75
ALTITUDE (Kft/Km)	30/9.1	70/21.3	90/27.4
GUIDANCE MODE	SINGLE MODE (NO INERTIAL MODE)	SINGLE OR DUAL MODE (NO INERTIAL MODE)	MULTI-MODE (IN-FLIGHT HANDOVER CAPABILITY)
AVIONICS INTERFACE	MINIMUM-GUIDANCE LOCK BEFORE LAUNCH, NO MISSILE PARAMETER INITIALIZATION	LIMITED MISSILE PARAMETER INITIALIZATION ALT, V_{MO}, CONTROL INITIALIZATION	UNLIMITED, BASED ON MISSILE PERF. REQTS.
MISSILE INSTRUMENTATION	SEEKER INSTRUMENTATION ONLY (SIMPLE AUTOPILOT)	SEEKER INSTRUMENTATION (2 RATE GYROS OR EQUIV) BODY INSTRUMENTS 3 RATE GYROS 3 ACCELEROMETERS	FULL SEEKER AND BODY INSTRUMENTATION (STRAPPED-DOWN INERTIAL REFERENCE SYSTEM)
TARGET SENSOR			
OPTICAL	IR RETICLE	IR-RETICLE OR ARRAY	IR-ARRAY
RADAR	SA-CW	SA-CW, SA-PD, ARM OR COMBINATIONS OF TWO.	CLASS II PLUS A-PD OR COMBINATIONS OF TWO OR THREE.

TABLE 2

MODULAR DIGITAL GUIDANCE SYSTEM FUNCTIONS

	MISSILE	
FUNCTION	CLASS I (MAX)	CLASS II (MIN)
Seeker	SA-CW	SA-PD
Signal Processing	64-Pt. FFT	64-Pt. FFT (Multiple Range Gates)
Head Control	Basic T&S (S1) Head Aim (S3)	Basic T&S (S1) Head Aim (S3)
Filtering and Estimation	Fixed Gain Filters (E1)	Switched Gain Filters (E2)
Guidance	Proportional Navigation (G1)	Proportional Navigational (G1)
Autopilot	Basic A/P (A1) Bandswitched Gains (A6)	Basic A/P (A1) Structural Filters & Fin Mixing (A2) Bandswitched Gains (A6)

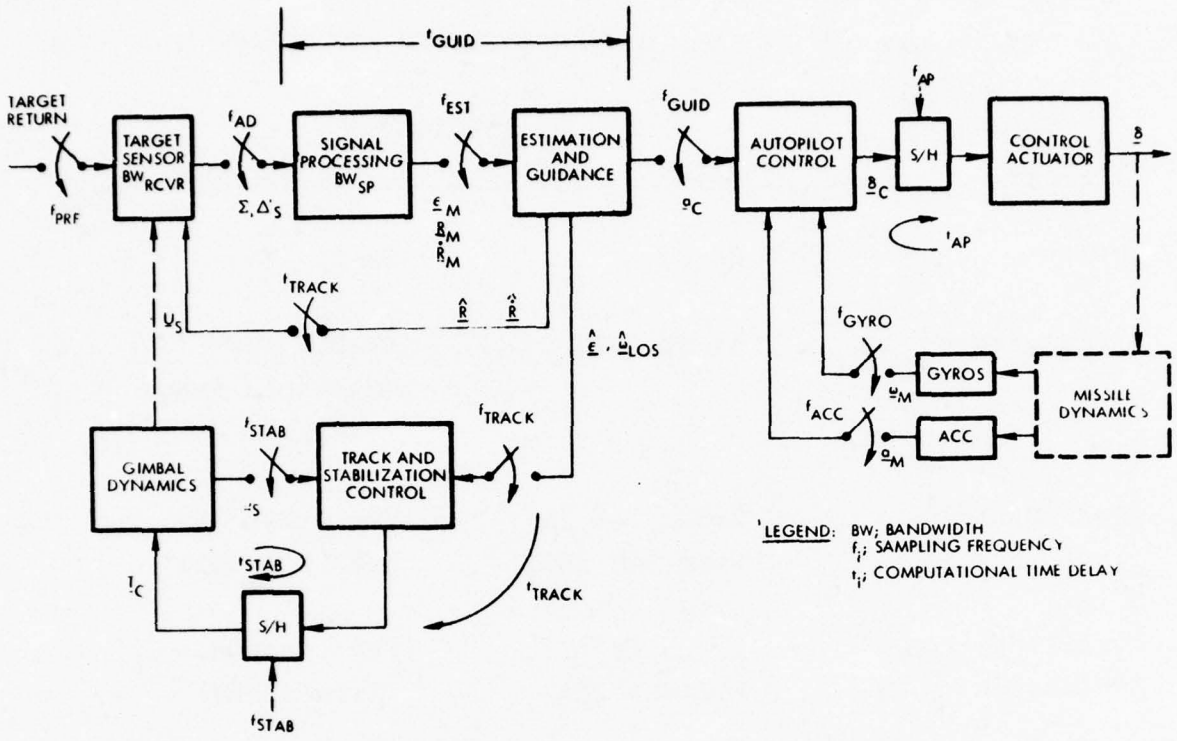


Figure 8
Digital Guidance System

Table 3 lists the corresponding data sampling and update rates for the two missile system models selected for this phase of the study program.

TABLE 3

DIGITAL GUIDANCE SYSTEM DESIGN PARAMETERS

CONTROL LOOP	MISSILE CLASS I (MAX)	MISSILE CLASS II (MIN)
Body Motion		
$f_{STAB}/f_{AP}/f_{GYRO}$ (Hz)	250	500
f_{ACC} (Hz)	125	250
A-D/D-A Conversion (Bits)	10	12
Computing Precision (Bits)	16	16
Steering/Guidance		
*f_{AD} Acquisition (KHz)	25	128
*f_{AD} Track (KHz)	20	32
A-D/D-A Conversion (Bits)	8	8
$f_{EST}/f_{GUID}/f_{TRACK}$ (Hz)	20	20
t_{GUID} (msec)	40	30
Computing Precision (Bits)	16	16

LEGEND: * Composite

3.1 Single and Federated Microcomputer System Considerations

During the Phase I and II studies the attributes and drawbacks of both single and federated microcomputer guidance systems were identified, (Ref. R-1 and R-2). Unlike common data base systems e.g. air defense, a missile guidance and control system readily partitions into four major, semi-autonomous functional groups, viz: seeker, head control, autopilot and fuze, with a 10 Hz interface between these groups resulting in a low I/O traffic situation.

A major objective of these studies has been to provide system design information and guidelines to enable a system designer to choose the optimum computer system design for the missile application in hand. Heretofore only the single computer system approach had been considered and implemented.

Figure 6 shows the system configurations resulting from single central and federated microcomputer design approaches respectively.

3.1.1 Single Computer Systems

Single computer guidance and control systems require a multi-wire analog and digital interface between all missile subsystems and the central computer. Similarly in the software, all program modules required to support the individual hardware subsystems must be time-multiplexed to maintain the data sampling rates and allowable computational delays for the various control loops which in turn results in a complex software module interface i.e. program linkages.

During a $1/f_{GUID}$ guidance command update interval say, 100 msec, both the seeker gimballed platform and autopilot functions must be computed 50 times, assuming a 2 msec update interval. Further, since the collection and processing of target seeker data to update the estimator and execute the guidance law requires approximately 80 msec, a "critical time slot" occurs during the remaining 20 msec of the 100 msec major interval. The latter timing situation is the throughput driver for single computer systems, since both the high-speed body motion functions, requiring 600-800 μsec max. computational delay (t_{STAB} and t_{AP}), and the steering command generation functions must be computed during this critical timing interval. Figure 9 is a time line of single computer processing activity for missile guidance and control. Estimated throughput requirements can approach 1.5 million operations per second (MOPS), which, with a 100% safety margin, requires a 3 MOPS computer to accommodate estimating errors and possible growth.

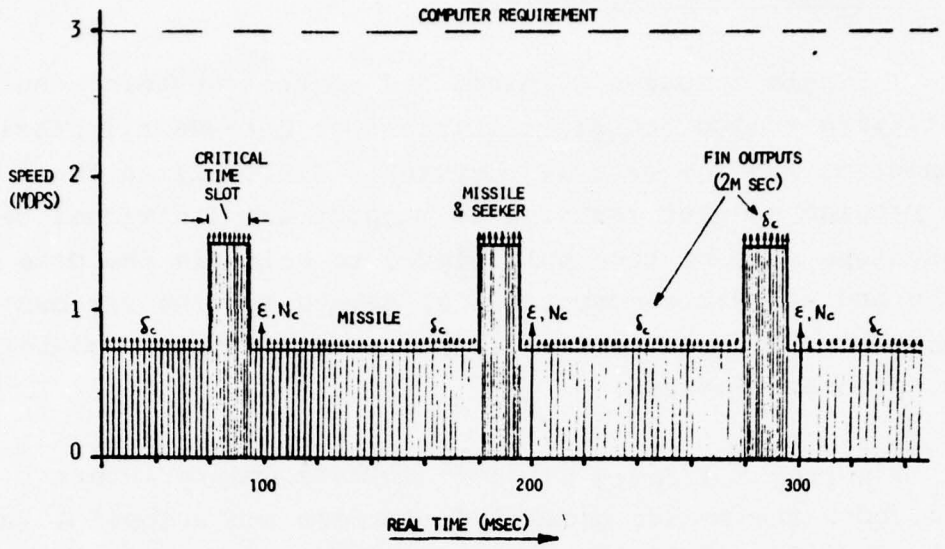


Figure 9
Single Computer Guidance and Control Throughput vs. Real-Time

A system designer should therefore take into consideration the above factors when making tradeoffs between single and federated microcomputer implementations of a given missile guidance and control system.

3.1.2 Federated Microcomputer Systems

By separating out the body motion stabilizing functions from the target seeker related functions and performing these tasks in separate microcomputers, the "critical time slot," high throughput situation of the single computer case is eliminated. The estimated throughput and memory requirements for these individual functions by missile class are as listed in Table 4 below.

It should be noted that the individual throughputs required fall within the capability of conventional general-purpose microcomputers currently on the market. Memory requirements are similarly reduced to more manageable proportions from a software design and development viewpoint.

Class I and II Missile System Designs - Based upon the general form and structure of digital missile guidance systems shown in Figure 8, two compatible federated microcomputer system designs were drawn up using the microcomputer macromodules defined in the Phase II Report. Figures 10 and 11 are block diagrams of the two missile systems i.e. Class I with a SA-CW radar seeker, and Class II with a SA-PD radar seeker.

TABLE 4

FEDERATED MICROCOMPUTER ESTIMATED THROUGHPUT & MEMORY REQUIREMENTS

MICROCOMPUTER	MISSILE CLASS I (MAX)		MISSILE CLASS II (MIN)	
	KOPS	WDS Prog./ Data	KOPS	WDS Prog./ Data
Steering Command Generation (Seeker)	91 +4.2 msec 64 Pt. FFT	1225/763	180 +0.84 msec 64 Pt. FFT	1287/3011
Gimballed Platform Stabilization	82	84/42	110	84/42
Autopilot	85	116/18	263	150/46
Fuze(Telemetry)	5(20)	46(50)/ 10 (53)	114(26)	228(50)/ 17(58)

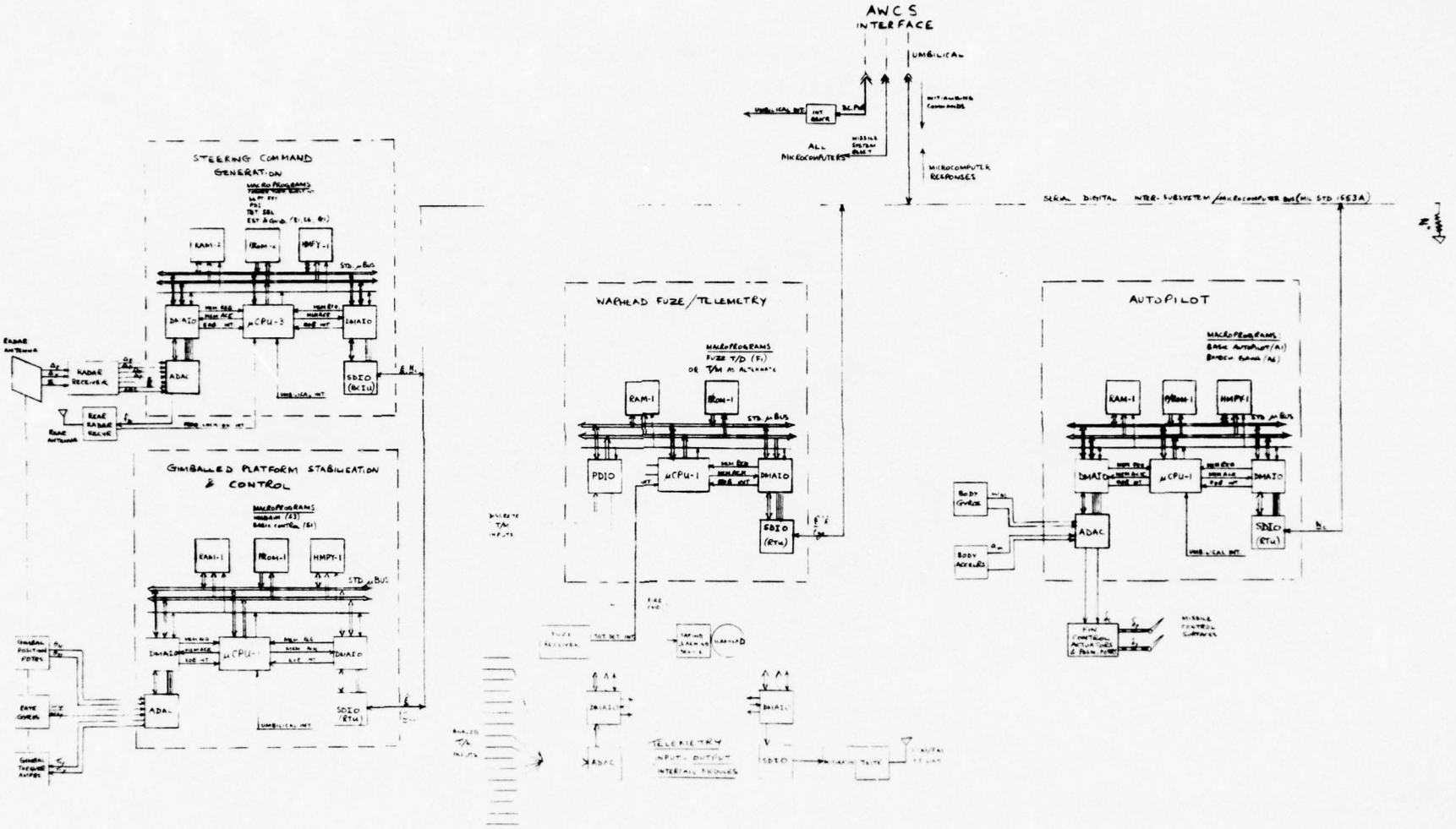


Figure 10 Macromodular Microcomputer System for Class I, SA-CW Radar Missile

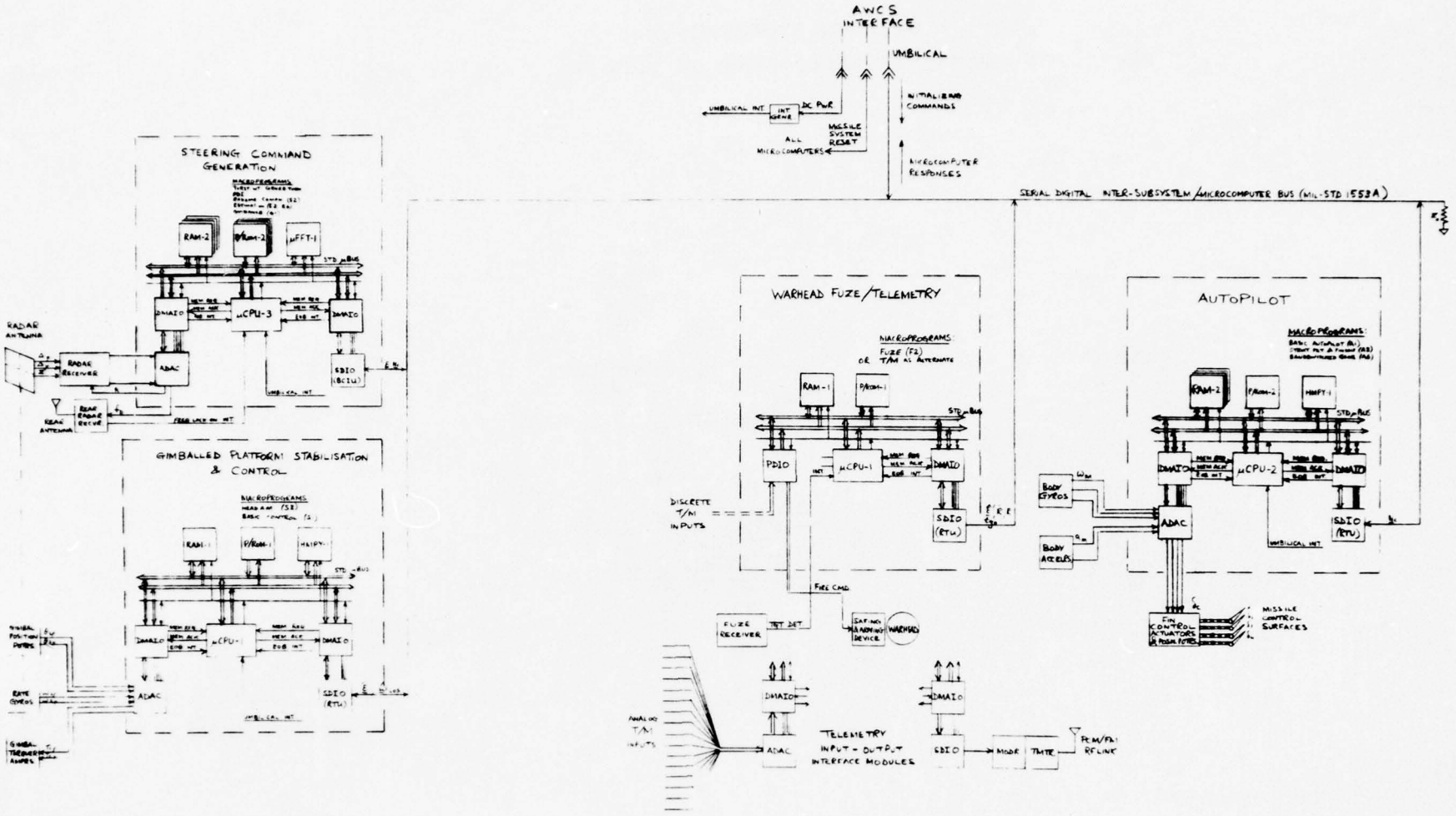


Figure 11 Macromodular Microcomputer System for Class II, SA-PD Radar Missile.

To achieve the modularity features of the federated systems without a severe cost penalty, μCPUs were selected to closely match the performance requirements of each subsystem. However, a two-computer seeker processing group (steering command generation), initially configured for the Class II SA-PD missile, was viewed as uneconomical. As a result of the computer performance work reported in Section 6, it was found that a single μBus μFFT module (64-pt.) incorporated into the estimation and guidance gp microcomputer, would enable the front-end FFT processor to be eliminated. Figure 11 reflects the revised system for the Class II missile. Hence, each class of missile system uses one 16-bit high-speed microprocessor and three low-cost 8-bit microprocessors.

Through this form of major function partitioning, compatibility is achieved with the following preferred modular design goals:

1. Design, development, manufacture, test and modification/updating of the seeker, platform, autopilot and fuze/telemetry subsystems as functional entities.

2. Standard 2-wire MIL-STD-1553A subsystem and avionics/umbilical interface.

3. Fault isolation reporting and repair at the subsystem level.

4. Design and development of new digital subsystems by traditional design specialist groups.

5. Enforcible software modularity through hardware interface.

The 1553A serial digital multiplex bus is used by the avionics weapons control system (AWCS) computer to access each subsystem via the umbilical before launch. The AWCS computer incorporates the 1553A bus control interface unit (BCIU) function, and all missile microcomputer SDIOs behave as slave units i.e. 1553A remote terminal units (RTUs). Upon commands from the AWCS, each subsystem in turn reports the results of its self-test programs which are executed immediately after D.C. power is supplied to the subsystems and associated microcomputers. Following the reporting of satisfactory self-test results to the AWCS, each subsystem microcomputer receives initializing data from the AWCS.

Missile launch is detected by each subsystem microcomputer through the occurrence of an umbilical interrupt, whereupon the body motion stabilization subsystem microcomputers (platform and autopilot) begin the cyclic process of: sampling co-located body motion sensors; the execution of control programs; and the outputting of compensating commands to the torquers/fin actuators. The steering command generation (SCG) subsystem microcomputer operates in a similar autonomous, closed-loop manner, in conjunction with the radar receiver, until the clutter and/or target is acquired and tracked. Thereafter missile and platform steering commands ($\underline{a}_c$ & $\underline{\varepsilon}$) are transferred from the SCG microcomputer's RAM module via DMAIO and SDIO interface modules to the RAMs of the autopilot and platform microcomputers respectively. This simple data transfer operation occurs every 100 msec, and results in an interrupt to the autopilot and platform μCPUs each time, to enable the cyclic stabilization programs to access an alternate I/O buffer area in RAM at the commencement of the next 2 msec update interval. The above process is repetitive throughout the flight.

The SCG microcomputer SDIO assumes the role of the AWCS computer BCIU after launch with all other subsystem SDIO modules operating as RTUs. The SCG microcomputer performs radar receiver mode control, signal processing, target selection, estimation and guidance functions, and, as the source of data for steering the missile and arming the fuze, is the logical master computer in the federated system.

Receipt of an arming command by the fuze microcomputer triggers the execution of the first part of the time delay algorithm. The detection of the target by the fuze receiver raises an interrupt line to the fuze μCPU. The time between arming and target detection is used to determine the optimum fuzing time delay following target detection, to achieve the highest kill probability for any given end game geometry.

More detailed descriptions covering the functions executed in each of the four microcomputers and the validation of their performance with respect to the system operational requirements, is given in Section 6 of this report.

4. MICROCOMPUTER MODULE DEFINITION

4.1 Introduction

Using the family of modules identified in the Phase II Report (Ref. R-2) as a point of departure, the further definition of a set of microcomputer macromodules for digital missile guidance and control applications was subject to an iterative development process involving the consideration of: software and system configuration requirements, circuit technology trends and preferred packaging methods.

For low-cost, flexible system implementations, which are the major goals of the studies, three factors had a strong influence upon the microcomputer module designs:

1. Use of existing support software for μCPU modules.

2. Use of high-volume, multi-source, microprocessor LSI/MSI circuit modules

3. Incorporation of FFT & PDI signal processing functions in a μBus module for direct μCPU control.

4. Use of a standard packaging method for all macromodules.

4.1.1 Software and CPU Architecture

The high cost of software and its dependence on ever increasing labor rates stressed the need to avoid the creation of

another computer with its own unique instruction set and support software package. Instead, the de facto standard Intel 8080A was adopted as the kernel processor for the two low-end microcomputers of the family, i.e. MIL temp. N-MOS and bipolar/CMOS/SOS RALU-based versions respectively, and, for Navy missiles, a microprocessor, RALU-based, version of the AN/UYK-20(V) central processing unit with a limited instruction set was assumed for the higher throughput 16-bit microcomputers in the family.

The 8080 is a very widely-used microprocessor with a mature support software package and, since the AN/UYK-20(V) is the Navy standard 16-bit minicomputer for shipboard applications with a software compatible counterpart (AN/AYK-14) for avionics systems, the logical extension of this common software philosophy of the Navy for the new generation of digital missiles, is a limited-instruction, microprocessor version of the AN/UYK-20/AYK-14 for digital missile guidance and control.

In other words, common support software and programming language for digital ships, planes and missiles.

Applications software for both the 8080-based and AN/UYK-20 microprocessors would be generated on either the regular full-blown counterparts of these machines observing the limited instruction sets of the target microprocessors or on suitable host computers, as preferred by the user.

4.1.2 LSI/MSI Circuit Technology

At the two military microprocessor standards workshops attended during this reporting period, (Refs. R-3 and R-4), the 8080 was recognized as a de-facto standard in the 8-bit,

CPU-on-a-chip, microprocessor field. The 4-bit slice RALU and associated microprogrammed control circuits, (i.e. AMD-2900 Series), were similarly recognized as the equivalent standard circuit modules for higher-speed, longer word length microprocessors. The instruction set for the latter being determined by the end user.

To achieve high-speed, low power consumption, small-size and radiation hardness, the CMOS/SOS (i.e. complimentary, metal-oxide semiconductor, silicon-on-sapphire), process was identified as the compatible circuit technology. The availability of functionally equivalent 2900-Series LSI/MSI modules in CMOS/SOS was viewed as highly desirable to meet the more stringent requirements of all space and military applications, thereby creating a volume market, common specifications and the necessary conditions for lower-cost CMOS/SOS.

4.1.3 FFT/PDI Signal Processing

In the Phase II report (Ref. R-2) medium and high-speed, microprogrammable, FFT processors were identified to perform pulse Doppler radar seeker signal processing in Class II and Class III missiles due to the lack of available throughput in general-purpose (gp) microprocessors. In the course of defining the Class II system model (Section 3) two computers, one FFT and one gp, were initially required to support the steering command generation function. Since the gp processor was required to control the front-end FFT processor and perform the remaining gp functions, the incorporation of FFT/PDI functions into an intelligent memory module for the gp processor provided a lower-cost and less complex alternative to the separate front-end FFT processor for missile applications. Further, this approach effectively closed the wide gap between the

available gp computer throughput and that of more special-purpose FFT processors designed for air defense and similar high throughput applications.

In terms of software, the new μFFT module would be the hardware counterpart of the software modules defined in the Phase II Report (Ref. R-2). Transition from a software to a hardware FFT implementation would then require the simple deletion of the software program modules, assuming a buffer area is used in memory to support the software version.

4.1.4 Standard Module Packaging

To maintain a standard μBus interface which all module designer builders, regardless of source, should meet to ensure the satisfactory integration of modules into a microcomputer by a system builder, the Navy standard electronic module (SEM) series was reviewed together with compatible hybrid LSI packages. As a result, it was found that the SEM-2A configuration would accommodate anyone of the macromodules (μCPU, RAM, ROM, HMPY, μFFT, PDIO, ADAC/DMAIO, SDIO/DMAIO) using fixed pin assignments on the 100-pin edge connector for system input/output, μBus and power inputs to the module. For a smaller form-factor, a double-sided SEM-1A would be compatible with fixed pin assignments for the μBus interface and module function-peculiar assignments for system input-output connections.

A standard 2" x 1" hybrid LSI package was selected for higher density packaging of the individual LSI/MSI CMOS/SOS microprocessor circuits, in the case of the high speed modules.

4.1.5 Microcomputer Modules

Table 5 is a revised list of 14 macromodules compatible with the SEM packaging standard. It should be noted that the DMAIO module is a hybrid-LSI module normally co-packaged with either the ADAC or SDIO, except for high-speed parallel digital I/O applications.

Figure 4 illustrates the physical form of the microcomputer SEMS and the standard hybrid-LSI package. Studies have shown that the number of circuits/chips packaged in hybrid form should not exceed between 30 and 40, depending upon the regularity of the interconnections, e.g. memory vs arithmetic and logical, in order to achieve reasonable yields in their manufacture. Packaging of an 8 or 16-bit μCPU macromodule in two hybrids, RALU and μPCU respectively, (Ref. R-2), mounted on a single SEM is only realistic using low-power CMOS/SOS technology in order to meet the power dissipation limitations.

TABLE 5

MICROCOMPUTER STANDARD ELECTRONIC MODULES (SEMs)

MICROPROCESSORS

SEM	DESCRIPTION	VLSI CIRCUIT	APPLICATION
1. μCPU-1	Microprocessor/Central Processing Unit, 8-Bit Byte General-Register, 2 μsec R-R Add	N-MOS, CPU-On-a-Chip, (MIL 8080)	o Telemetry o Fuzing o Head o Autopilot
2. μCPU-2	Microprocessor/Central Processing Unit, 8-Bit Byte, General-Register 600 nsec (8080 Emulator)	CMOS-SOS, Bit-Slice RALU & μPCU Hybrids (2900/3000-Series or Equiv.)	o Autopilot o Head Control o Fuzing
3. μCPU-3	Microprocessor/Central Processing Unit, 16-Bit Word, Fixed-Point, General-Register, 600 nsec R-R Add	CMOS-SOS, Bit-Slice RALU & μPCU Hybrids (2900/3000-Series or Equiv.)	o Autopilot (Adaptive)
4. μCPU-4	Microprocessor/Central Unit, 16-Bit Word, Fixed & Floating-Point, General-Register, 600 nsec R-R Add (1.0 to 3.25 μsec Flt. Pt.)	CMOS-SOS, Bit-Slice RALU & μPCU Hybrids (2900/3000-Series or Equiv.)	o Signal Processing o Estimation o Guidance

TABLE 5 (CONT.)

HIGH SPEED ARITHMETIC

SEM	DESCRIPTION	VLSI CIRCUIT TECHNOLOGY	APPLICATION
5. HMPY-1	Hardware Multiplier, 200 nsec, 16 x 16-Bit Multiply	CMOS-SOS Single Hybrid	o Throughput Enhancement for μCPU e.g. Class I Sig. Proc.
6. μFFT-1	Micro Fast-Fourier Transform Processor, 40-400 μsec 64-Pts, 8 + J8.	CMOS-SOS or CCD RALU & μPCU Hybrids (2900-Series or Equiv.)	o Throughput Enhancement for μCPUs e.g. Class II & III Sig. Proc.

TABLE 5 (CONT.)

MEMORIES

SEM	DESCRIPTION	VLSI CIRCUIT TECHNOLOGY	APPLICATION
7. RAM-1	Random-Access, Read/ Write Memory, Medium-Speed, 128-2K Bytes, 500 nsec Max. Access Time	N-MOS Dip/Hybrid	Data o Autopilot o Head Control o Telemetry o Fuzing
8. P/ROM-1	Programmable (Mask/ Electrically) Read-Only Memory, Medium-Speed, 1K-16K Bytes, 500 nsec Max. Access Time	N-MOS Dip/Hybrid	Programs o Autopilot o Head Control o Telemetry o Fuzing
9. RAM-2	RANDOM-ACCESS, Read/ Write Memory, High-Speed, 256-1K x 16-Bit or 256-2K Bytes, 100 nsec Max. Access Time	CMOS-SOS Dip/Hybrid	Data o Sig. Proc. o Estimation o Guidance o Head Control o Autopilot o Fuzing
10. P/ROM-2	Programmable (Mask/ Electrically) Read-Only Memory, High-Speed, 1K-4K x 16-Bits or 1K-8K Bytes, 100 nsec Max. Access Time	CMOS/SOS Dip/Hybrid	Programs o Sig. Proc. o Estimation o Guidance o Head Control o Autopilot o Fuzing

TABLE 5 (CONT.)

INPUT-OUTPUT MODULES

SEM	DESCRIPTION	VLSI CIRCUIT TECHNOLOGY	APPLICATION
11. DMAIO	Direct Memory Access Input-Output Channel, Parallel Word/Byte Transfers To/From Microcomputer RAM	CMOS-SOS/Bipolar Single Hybrid	All Microprocessor Applications
12. PDIO	Parallel Digital Input-Output Channel, Parallel Discrete Transfers To/From μCPU	CMOS-SOS/Bipolar Single Hybrid	o Telemetry o Fuzing
13. ADAC	Analog To Digital/ Digital To Analog Input-Output Channel A-D: 8/16/24 Chs.,Sim. S/H, Mux, 8/10/12-Bit,A-D 3/6/8 μsec Max/Ch. D-A: 8 Chs. Demux., S/H, 12-Bit D-A, 5 μsec Max/Ch.	CMOS-SOS Single Hybrid	o Head Control o Autopilot o Telemetry o Radar Receiver
14. SDIO	Serial Digital Input-Output Channel Word & Bit Serial Data/Command Transfers, 1Mbit/sec Max, MIL-STD-1553A	CMOS-SOS Single Hybrid	o Avionics o Inter-Microcomputer

4.1.6 Microcomputer Configurations

Figure 12 shows four microcomputer configurations covering the range of throughput requirements derived in the Phase II study for the three missile classes. Throughput figures for the 8-bit processors are equivalent for 16-bit precision arithmetic operations.

Low-Cost 8-Bit - The low-cost 8-bit 8080-based microcomputer meets the throughput requirements for either telemetry, fuzing, gimballed platform control and autopilot, (the latter two functions with a HMPY module), for a Class I missile, as described in Section 6. Since program sizes are small (1K bytes approx.), (See Section 6), 8-bit processors are practical for these single-function applications.

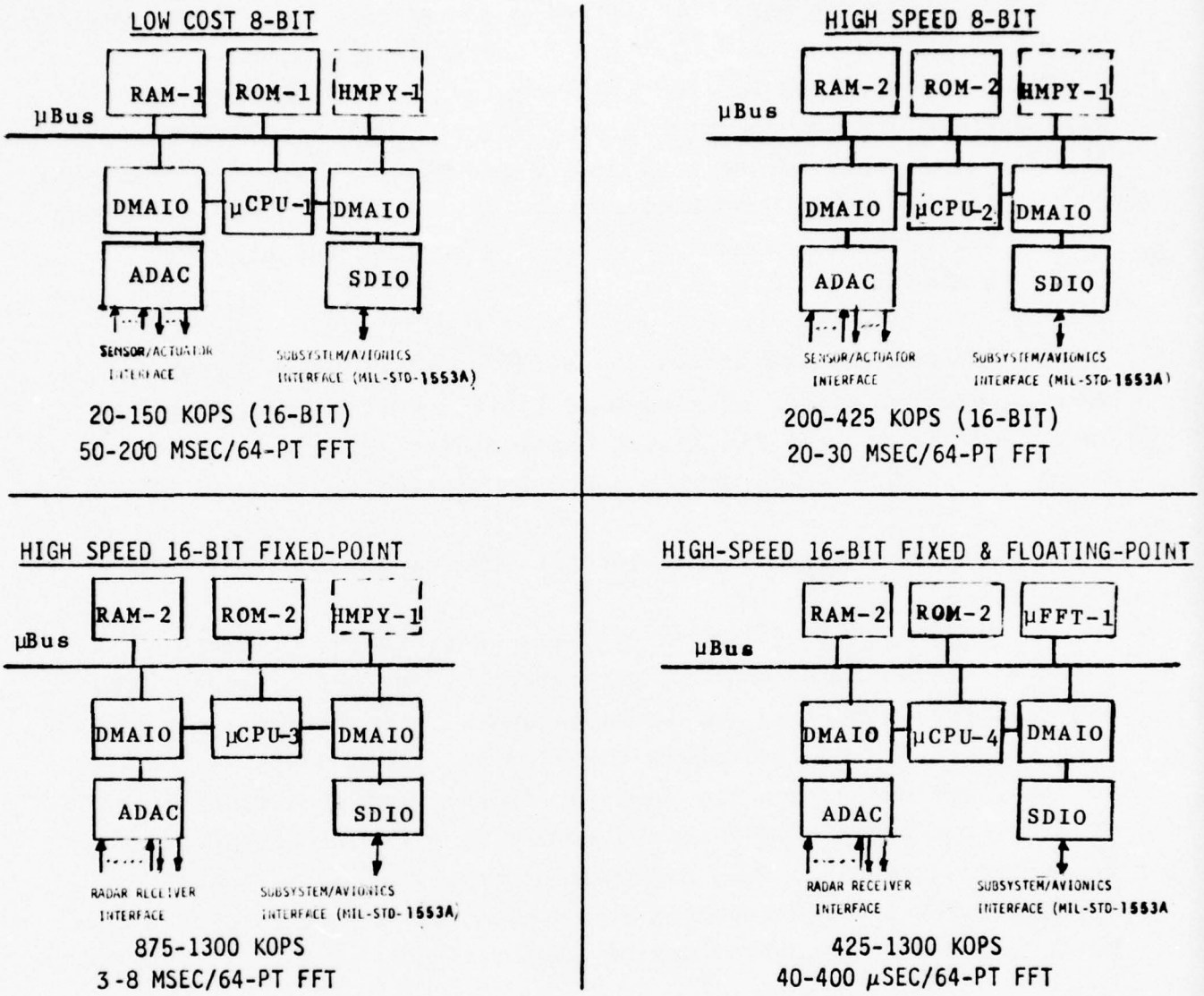


Figure 12

Macromodular Microcomputer Configurations

High-Speed 8-Bit - By replacing the μCPU-1 wih a CMOS/SOS bit-slice equivalent, i.e. per the Signetics and AMD emulator designs, (Refs. 8 and 9), and the RAM-1 and ROM-1 modules with similar high-speed circuit equivalents, (RAM-2 and ROM-2), the throughput is increased approximately 3 to 10 fold depending on instruction mix. Such a microcomputer is suitable for executing the higher performance head or autopilot control function in Class II missiles.

Similar modular growth is achieved in software, using the same modules coded for the low-cost 8-bit computer and adding the structural filter and fin mixing modules (A2) for the Class II autopilot, per the flow charts given in Section 6.

The I/O remains intact for this system update.

High-Speed 16-Bit Fixed-Point - For Class I guidance processing, (steering command generation), high-performance Class III adaptive autopilots, or in cases where a single high-performance computer satisfies the missile system design constraints, and, where the need for floating-point arithmetic is minimal, the 8-bit μCPU-2 is replaced with a 16-bit fixed-point processor, (μCPU-3). This macromodule substitution achieves approximately a 4:1 throughput improvement at the expense of higher parts count and associated design complexity.

Again, the I/O modules remain unchanged.

High-Speed 16-Bit Fixed and Floating-Point - For Class II (max) and Class III missiles employing Kalman filters for state estimation, (program modules E3 and E4), floating-point arithmetic becomes a distinct advantage in the coding process. By including

microprograms in the μPCU module for floating-point arithmetic operations μCPU-3 becomes μCPU-4. Also, to satisfy the multiple range gate FFT processing load, the μFFT-1 module is added to the μBus, to replace the 64-point FFT program modules. General-purpose throughput depends upon the use of a hardware (HMPY-1) or firmware (microcode) multiply capability, as indicated in the throughputs shown.

Input-output interface with the microcomputer remains unchanged and supported by the same macromodules.

Figure 13 plots respective gp throughputs of each of the four basic microcomputer configurations with and without the hardware multiply module.

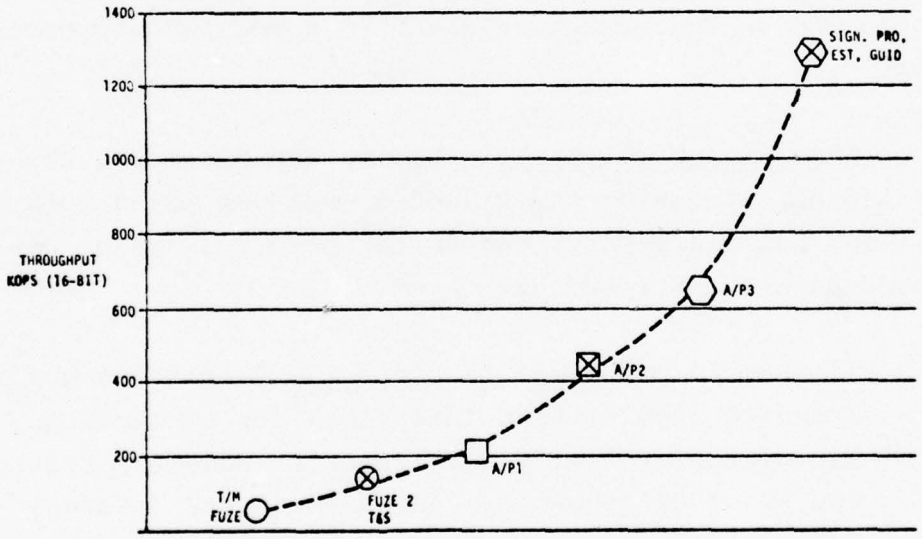


Figure 13
Macromodular Microcomputer Throughputs

4.2 Module Functional Descriptions

In this subsection the theory of operation of each module is described, supported by functional block diagrams, state transition diagrams, and timing diagrams where significant.

4.2.1 μBus Definition

Of the many different types of microcomputers currently available on the market, most of these use a parallel digital bus of one form or another to interface the microprocessor to the memory and input-output circuits. However, although these bus configurations are similar, they all differ in small detail such that the memory and I/O circuits of one type will not readily interface with the microprocessor of another. This fact was recognized in the Phase II study and a standard microbus was recommended as the crucial element in a modular microcomputer family.

μBus Design Rationale - During the Phase III study, the detailed definition of the standard μBus was based upon the following rationale pertinent to the practicality of the bus and system operational requirements:

1. μBus Standarization Concept - Determines and maintains standard input-output interfaces for connecting microcomputer family macrofunction modules, regardless of the phase of technology advancement and internal improvements/changes made to the modules.

2. μBus Use/Extent - Internal, (12 inches. max length), parallel digital interface between microcomputer macrofunction modules (i.e. microprocessor (μCPU), memory (RAMs & P/ROMs) high-speed arithmetic (μFFT, HMPY) and I/O (DMAIO & PDIO)) for a simplex microcomputer configuration. Federated microcomputer systems to interface between individual microcomputer I/O channels.

3. μBus Traffic - Minimized by autonomy of user modules through architectures which minimize frequent memory accesses/overhead operations.

4. μBus Control - μCPU receives memory access requests from DMAIO modules and provides access to μbus subject to completion of current μCPU - memory data/instruction transfer. Priority of access assigned by system timing constraints. DMAIO module generates end-of-block (EOB) interrupt for μCPU upon completion of data transfers to memory.

In support of item 3 above, the following functional characteristics of the interfacing macromodules are required:

1. μCPU Module Architecture - General-register for active and partial results, using multi-address, register-register instructions. No programmed I/O transfers to/from memory via μCPU except initializing commands to I/O channels.

2. μFFT Module Architecture - Internal microprogram control and register file to store FFT data points for high-speed/pipelined, butterfly arithmetic unit operation.

3. I/O Modules - Internal microprogram control. Can initiate input sampling and data transfers to main memory in response to integral cyclic interval timers i.e. for repetitive body motion sensing and stability loop processing. Can be commanded, (by programmed instructions), to initiate data transfers from memory for conversion and output to gimbal torquers/fin actuators. Radar sensor data sampling can be initiated by a programmed command and sampling rates similarly programmed.

Semi-autonomous microprogram control becomes a key constituent of each module, (with the exception of RAMs and ROMs), to achieve low bus traffic and a reduction in software. This shift from software to hardware control is supported by current microprocessor LSI/MSI circuit technology.

μBus Interface Lines - Since all μBus modules are treated as memory modules by the μCPU, a practical memory interface drives the bus configuration.

With the advent of 8-bit micro processors, byte organized semiconductor RAMs and (P)ROMs have appeared on the market, and with these new memory-on-a-chip modules the provision of multiple "chip select" lines.

This contrasts with the earlier 1K x 1 Bit RAM chip organizations where a user would configure a byte or word organized memory with several of these 1 bit wide devices.

Based upon the above technology trend, the new memory-on-a-chip LSI circuits were used to establish standard μBus interface lines. Figure 14 illustrates the resulting interface arrangement between μCPU, memory and I/O modules. Module outputs to the μBus are via tri-state drivers, to present a high source impedance when inactive/disabled. System reset, clock and timing reference inputs for modules other than memories, are shown in subsequent module interface diagrams.

Adopting the semiconductor memory interface as a standard obviates the need to re-package RAMs and (P)ROMs and allows modular expansion of memory on a dual in-line package (DIP) basis. Further, chip select lines become the equivalent of module select lines.

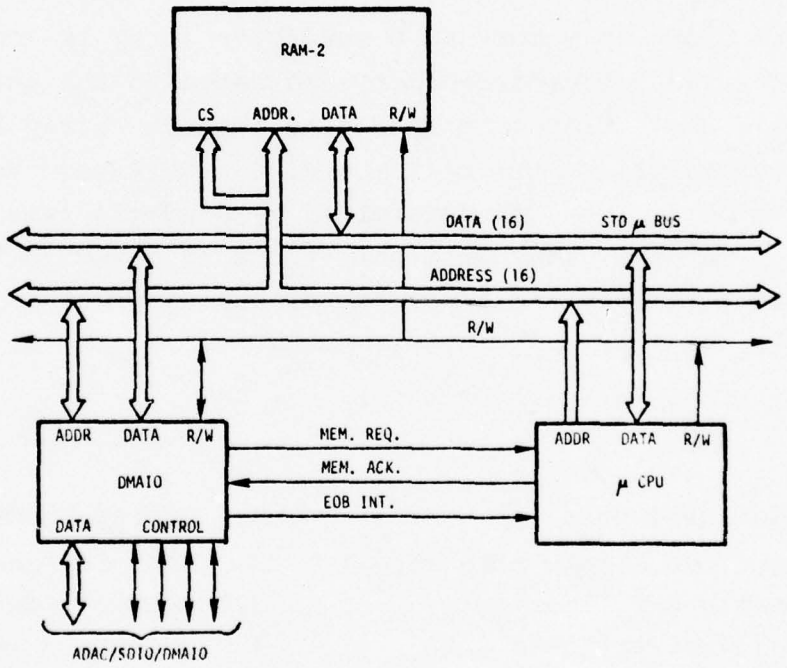


Figure 14
Macromodular Microcomputer Standard Interfaces

In the case of 8-bit μCPUs, only the 8 most significant data lines are used.

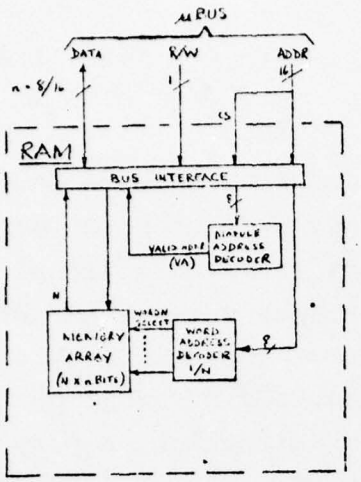
4.2.2 RAM and (P)ROM Modules

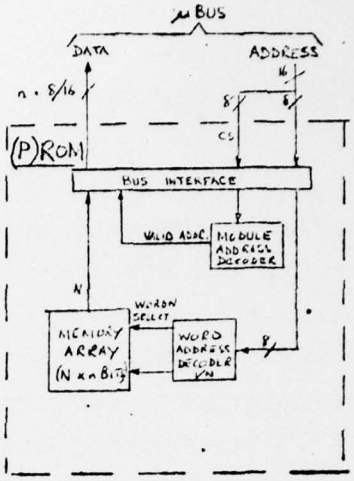
Since all μBus macromodules appear as memory modules to a μCPU, the definition of the μBus interface lines and the associated timing of data address and control signals was based upon current microprocessor RAM and (P)ROM timing and interface.

Semiconductor RAM and (P)ROM Configurations - Figure 15 (A) and (B) show the functional configurations and associated timing diagrams of typical semiconductor RAMs and (P)ROMs respectivley.

In both types of memory an N word/byte array is addressed via a 1/N decoder, and a second decoder, dedicated to the chip select lines, controls the gating of stable data into the array and/or out of the array via a tri-state, bidirectional interface. The choice of read/write for RAMs is determined by a single read/write (R/W) line. The individual internal delays involved in the process are discussed in Ref. R-11, but compatible external timing sequences are as follows:

RAM	(P) ROM
1. Address module	1. Address module
2. Wait for access time (t_{acc})	2. Wait for access time (t_{acc})
3. Enable R/W line	3. Read-out data
4. Read-out/write-in data	4. Do not re-initiate sequence until end of cycle time (t_{cyc})
5. Do not re-initiate sequence until end of cycle time (t_{cyc})	



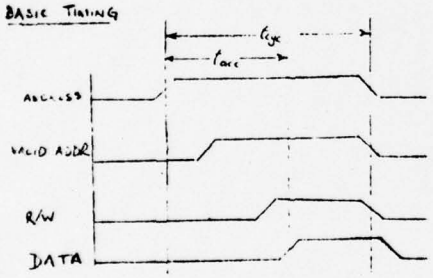


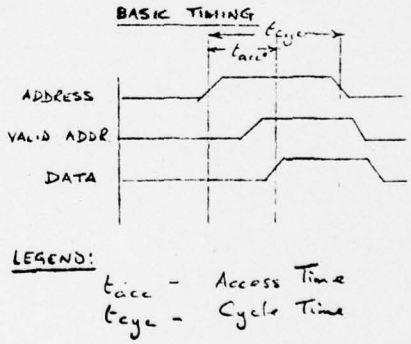
MEMORY TYPES & PERFORMANCE

Interval	RAM-1	RAM-2
t_{acc} (nsec)	500	100
t_{cyc} (nsec)	1000	200

MEMORY TYPES & PERFORMANCE

Interval	(P)ROM-1	(P)ROM-2
t_{acc} (nsec)	500	100
t_{cyc} (nsec)	1000	200





(A) RAM

(B) (P)ROM

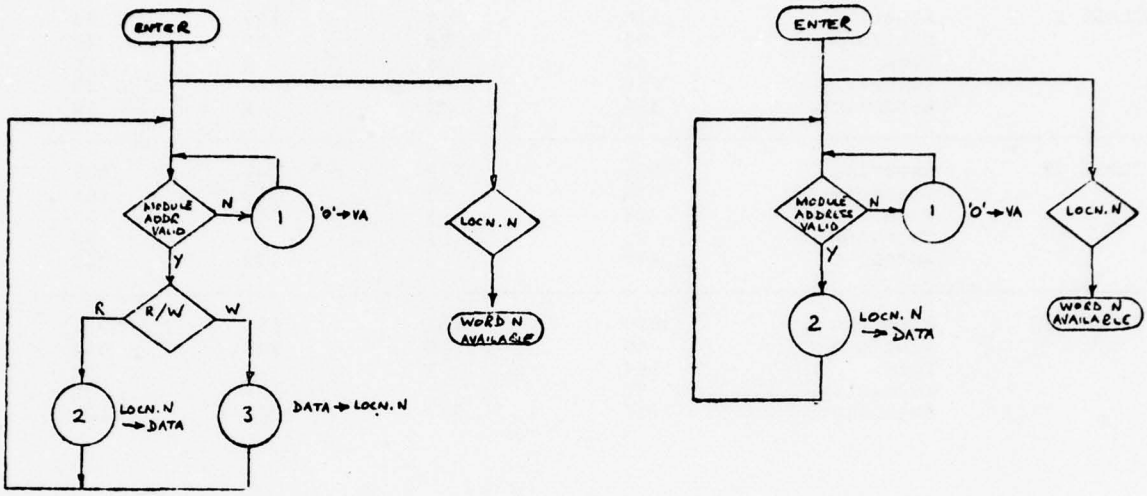
Figure 15

RAM and (P)ROM Module Functional Block Diagrams and Timing Sequences.

Figure 16 (A) and (B) shows the equivalent state transition diagrams for the two memory types.

RAM and (P)ROM Synchronizing - Compared to core and large maxi/minicomputer memory systems, the new semiconductor memory-on-a-chip LSI devices provide no data/memory available signal with which to synchronize the transfer of data between a RAM/(P)ROM and a µCPU or DMAIO module. To provide direct compatibility with these new microprocessor memory modules without adding additional timing circuits, thereby causing a re-packaging situation, a pre-programmed time delay is incorporated in the µCPU and DMAIO modules respectively to generate a pseudo data/memory available signal based upon the speed of the memory module being employed i.e. RAM/(P)ROM-1 or RAM/(P)ROM-2.

The timing of staggered addressed and R/W signals is based upon a 2-phase microcprocessor clock.



(A) RAM

(B) (P)ROM

Figure 16
RAM and (P)ROM State Transition Diagrams

Memory/Module Addressing - A review of missile memory requirements given in the Phase II Report shows that a 256-word memory module provides a reasonable basic building block for the three missile classes. Table 6 lists the memory requirements by missile class, major function and use (program, volatile data and constants). Table 7 translates these data into maximum and minimum requirements for seeker processing (steering command generation) and all other functions, with the resulting range in number of 256-word modules required in each case.

TABLE 6

MEMORY ADDRESSING REQUIREMENTS

MISSILE	FUNCTION	PROGRAM (ROM)	MEMORY (WDS) DATA (RAM)	DATA (ROM)	(TOTAL)
CLASS I	Steering	1170	605	190	795
	Platform	84	26	16	42
	Fuze	46	6	4	10
	Telemetry	50	53	-	53
	Autopilot	116	10	8	18
CLASS II	Steering	1935	2858	150	3008
	Platform	272	50	359	409
	Fuze	228	5	12	17
	Telemetry	50	58	-	58
	Autopilot	830	173	179	352
CLASS III	Steering	3939	5621	150	5771
	Platform	449	62	359	421
	Fuze	184	3	7	10
	Telemetry	50	71	-	71
	Autopilot	1716	207	1280	1487

TABLE 7

MISSILE MEMORY MODULE REQUIREMENTS

(A) STEERING COMMAND GENERATION

STORAGE REQT.	MAX	MIN	COMMON MODULAR INCREMENT	NO. OF MODULES
Program	3939	1170	256	5-16
Volatile Data	5621	605	256	3-2
Constants	190	150	256	1

(B) OTHER FUNCTIONS

	MAX	MIN	COMMON MODULAR INCREMENT	NO. OF MODULES
Program	1716	46	256	1-7
Volatile Data	207	3	256	2
Constants	1280	4	256	1-5

Relating the above data to currently available semiconductor memory package configurations reveals the situation shown in Table 8 ROMs and (P)ROMs are widely available in 8-bit organizations and in MOS and bipolar technologies, RAMs however are still bit organized in bipolar and few are available in MOS technology. This situation could be expected to change as the demand for higher-speed microprocessors increases, and CMOS/SOS is used as the high-speed, low-power alternative to bipolar for higher packing-densities.

Addressing Scheme - Using the 256-word module as a basic building block, a suitable addressing scheme, using the 16 μBus address lines, would be as shown in Figure 17.

TABLE 8

STANDARD INDUSTRY SEMICONDUCTOR MEMORY MODULES/PACKAGES
& CURRENTLY AVAILABLE PACKING DENSITIES

	MOS			BIPOLAR		
TYPE	ORGANI-ZATION	ACCESS (nsec)	INDUSTRY PART	ORGANI-ZATION	ACCESS (nsec)	INDUSTRY PART
	128 x 8	350	MCM6810ALI	256 x 1	55	Am27LS00XM
RAMS	256 x 4	300	Am91LO1C	1024 x 1	100	93415
	1024 x 4	300	Am9130C			
(STATIC)	*1024 x 4	100	MWS5114			
	*256 x 4	90	MWS5540			
	*128 x 8	200	MWS5080			
PROMs	256 x 8	750	Am9702-1	32 x 8	75	Am27LS09XM
			Erasable	256 x 8	85	MMI5309-1
				512 x 8	85	MMI5249-1
				1024 x 8	125	MMI5387-1
ROMs	512 x 8	500	Am9214	32 x 8	60	MMI5231-1
	1024 x 8	400	Am9208B	256 x 4	60	MMI5201-1
	2048 x 8	300	Am9216C	512 x 8	90	MMI5241-1
				1024 x 8	275	Am27581XM
				2048 x 8	120	MMI5276

NOTES:

* MWS5000 - RCA CMOS/SOS
MCM - Motorola
Am - Advanced Micro Devices
MMI - Monolithic Memories
All with tri-state outputs.

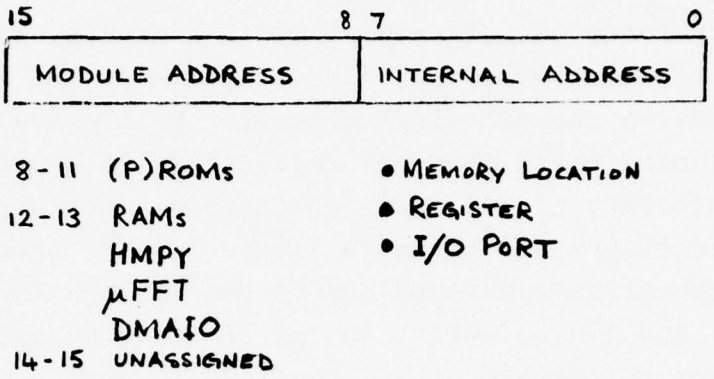


Figure 17
Module Addressing Scheme

4.2.3 μCPU Module Definitions

The four microprocessor macromodules identified for digital missile guidance and control applications i.e. μCPU-1, -2, -3 and -4, utilize existing processor architectures and instruction sets, (either complete or a subset), to achieve software compatibility with already mature and widely used machines viz: Intel 8080A and Navy AN/UYK-20/AYK-14.

The primary task in this portion of the study was the definition of the interface logic necessary to achieve compatibility between the above types of processor and the other μBus macromodules. State transition and timing diagrams were

generated to establish the fundamental timing and integrity of these processors in relation to the modular family.

μCPU-1 - Figure 18 shows the kernel 8080A CPU-on-a-chip microprocessor incorporated into the μCPU-1 module, with clock, power-on/auto reset, memory synchronizing, DMAIO and μBus interface circuits added. A characteristics profile of the 8080 is given in Section 6. Timing state diagrams reflecting the operation of the additional interface circuitry in response to external commands and 8080 operating status are shown in Figure 19. Further details of the 8080 operation are contained in the Systems User's Manual (Ref. R-12). The timing relationships of the 8080 and μBus data, address and control waveforms are shown in Figure 20.

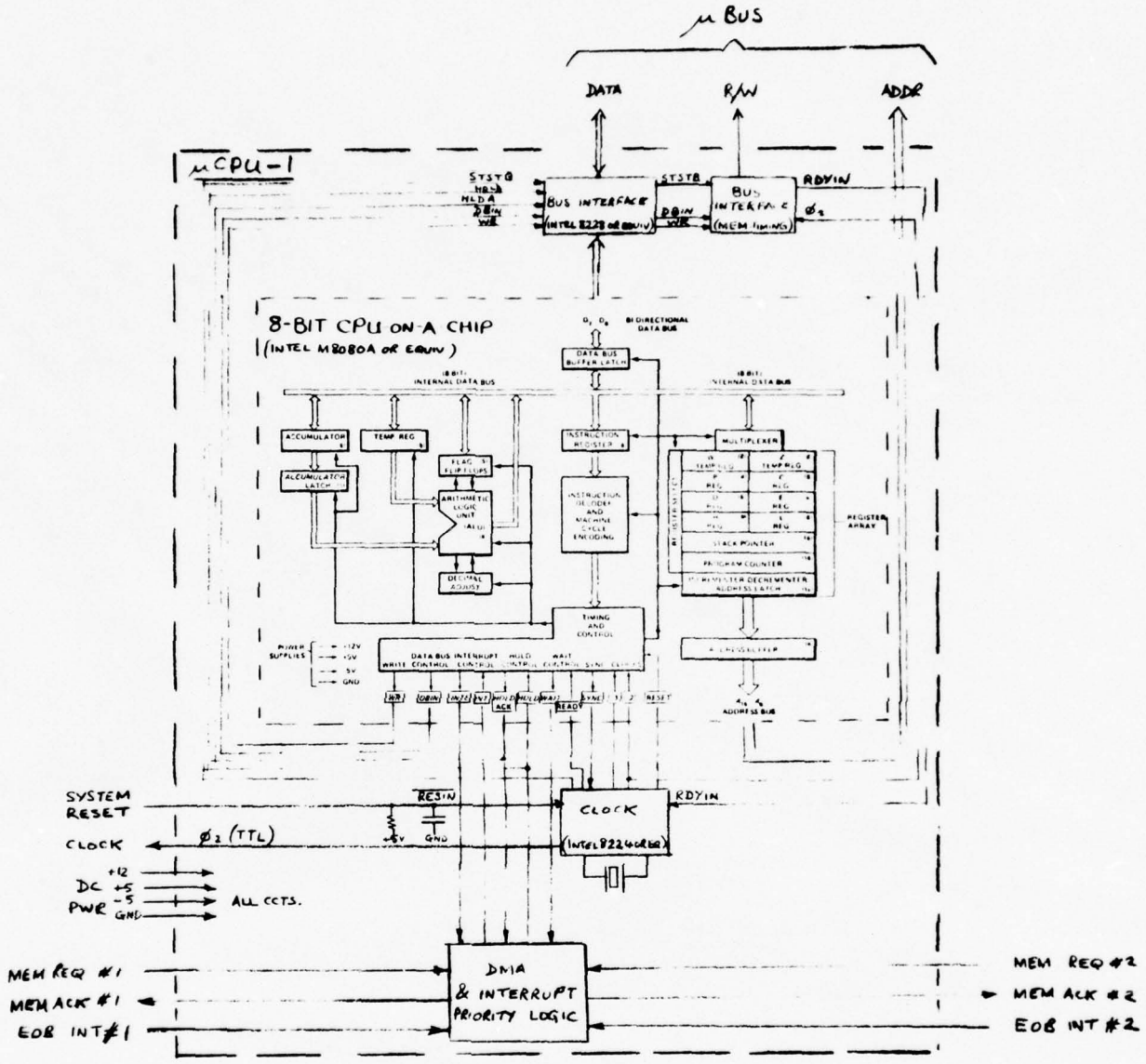


Figure 18

8080-Based μCPU-1 Block Diagram

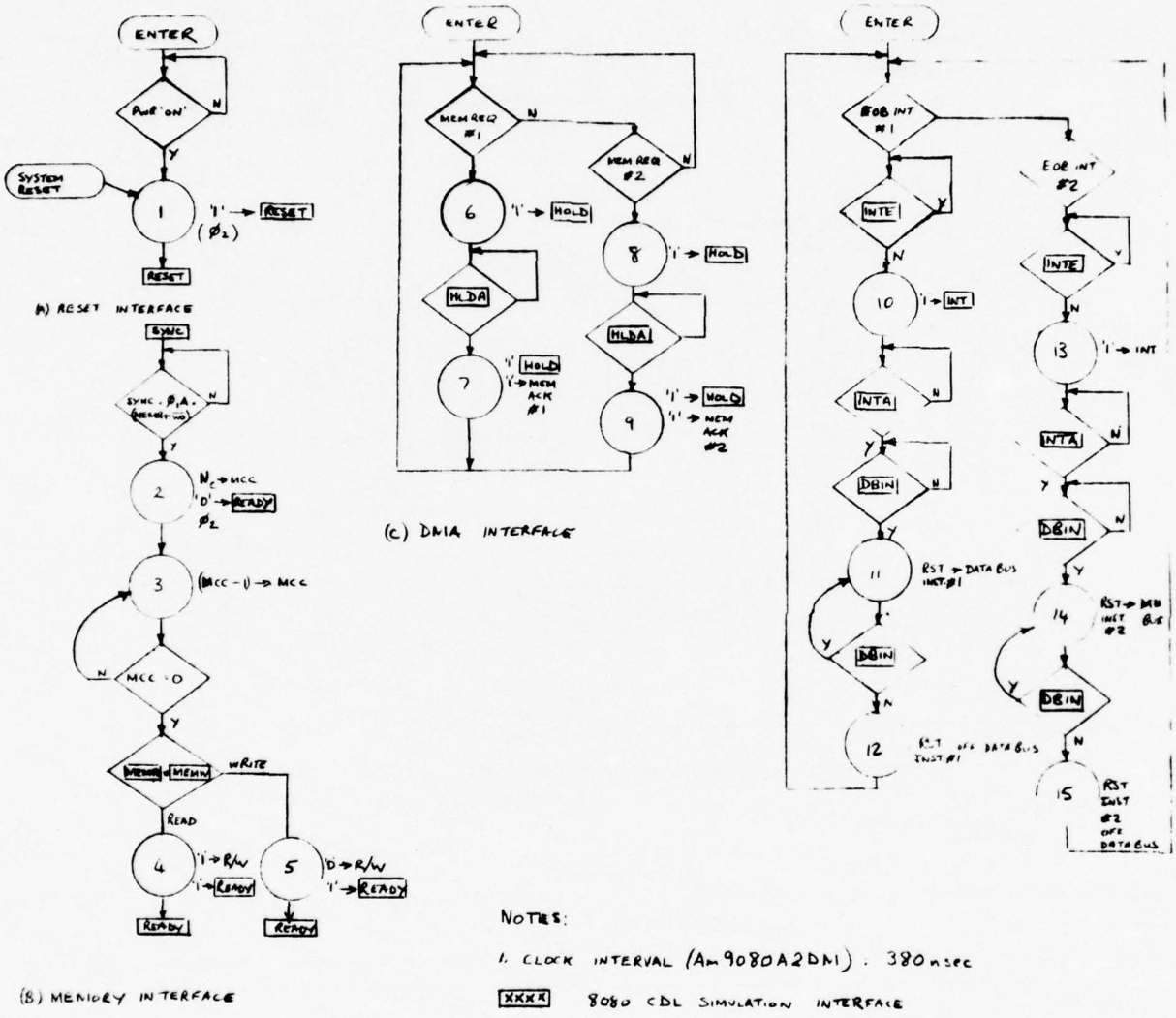


Figure 19

μCPU-1 8080 Interface State Transition Diagrams

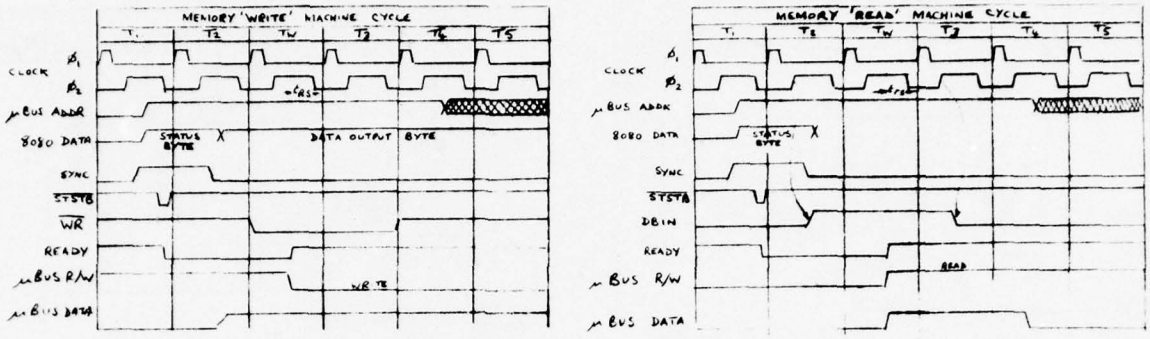
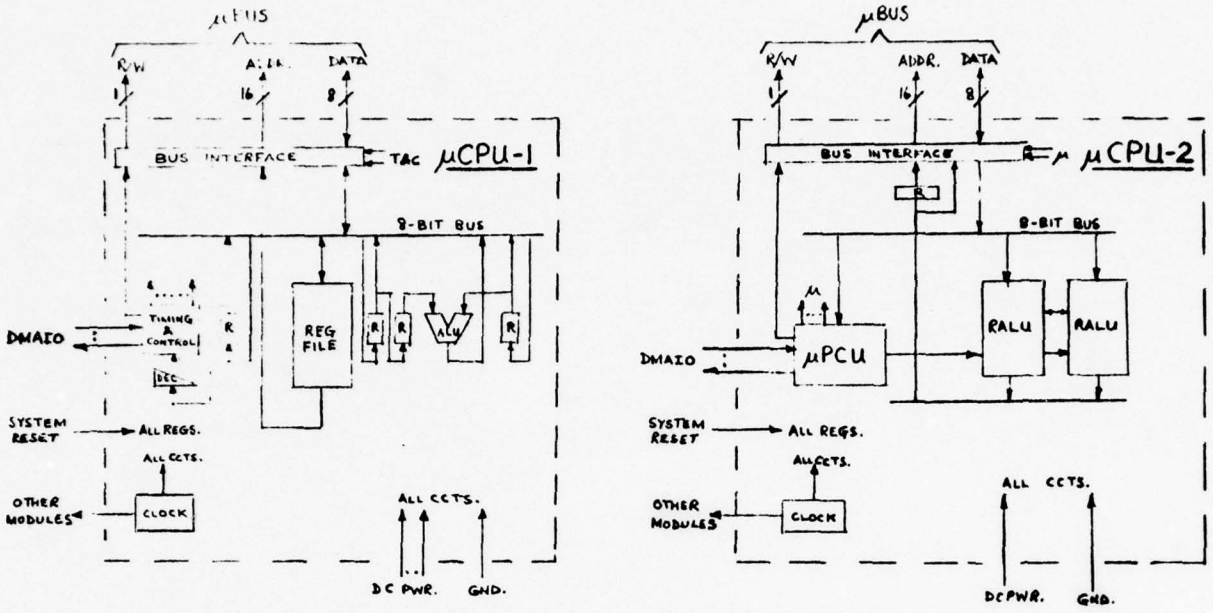


Figure 20

μCPU-1, 8080/μBus Interface Waveforms

The μCPU-1 clock is output to all other modules in a given microcomputer configuration except the RAMs and (P)ROMs.

μCPU-2 - This module is functionally equivalent to the N-MOS 8080A-based μCPU-1, but is implemented with high-speed bit-slice, CMOS/SOS RALU circuits and an associated microprogram control unit (μPCU). Figure 21 illustrates the transition from μCPU-1 to μCPU-2.



(A) μCPU-1 (B) μCPU-2

Figure 21

μCPU-1 and μCPU-2 RALU Equivalent, Block Diagrams

It was originally intended to use a hardwired control unit (HWCU-1) for the bit-slice 8080 emulator, (Ref. R-2), since the instruction set could be assumed to be fixed. However, the recent availability of low-cost, LSI/MSI microprogam control circuits as part of a modular family of LSI/MSI devices, tends to offset the small hardware savings accrued from the less flexible HWCU approach. Examples of the former approach are evidenced in the Signetics 3000-Series and AMD 2900-Series 8080 emulators, (Refs. 8 and 9).

μ CPU-3 and μ CPU-4 - These processor modules are bit-slice, CMOS/SOS RALU and μPCU implementations of the Navy AN/UYK-20(V) with a limited instruction set sufficient for the needs of digital missile guidance and control. Architectural characteristics are given in Section 6 and futher details concerning the operation of the processor are covered in the user's manuals, (Ref. R-13 and R-14). As in the case of the 8080-based μCPU-1, the necessary interface circuits are provided for compatibility with other μBus macromodules. Programmed I/O instructions are not used. Figure 22 is a simplified block diagram of the μCPU-3 and -4, processor configurations, to illustrate the fundamental similarity between these 16-bit microprocessors and their 8-Bit counterpart μCPU-2.

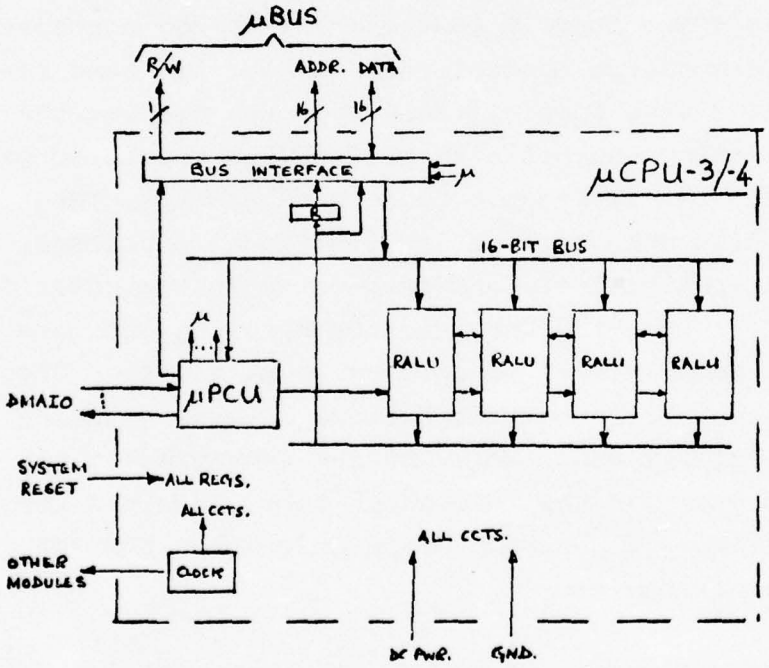


Figure 22
μCPU-3 and -4 Simplified Block Diagrams

Figure 23 is a state transition diagram representative of 16-bit general-register processors with multiple operand addressing modes, e.g. Digital Equipment Corp: PDP-11/34 and Navy AN/AYK-14. Definitions of these modes are given in Appendix A. μCPU-4 is identical to μCPU-3 with microprograms added for executing floating-point arithmetic instructions.

4.2.4 μFFT Module Definitions

To provide the necessary throughput boost for high-speed radar sensor signal processing within a gp microcomputer configuration, as opposed to the provision of an additional "front-end" signal processor, the Phase II signal processing CPU incorporating FFT butterfly module (μFFT-1) and associated memory (RAM) and microprogram control unit (μPCU), has been re-configured as a RALU-based μBus module. This approach enables the general-purpose processing capabilities of the μCPU module to be used for the relatively low throughput radar mode control, target selection, state estimation and guidance law functions and closely integrated with the spectrum analysis and post-detection integration functions. It also provides a better match of hardware to missile-type radar signal processing requirements. Figure 24 illustrates the throughput disparities between existing very high-speed front-end FFT processors, of the type used in air defense and avionics systems, and the throughput rates required for Class I, II and III missiles. The common 64-point complex FFT was used as the benchmark in all cases.

RALU-Based μFFT - Figure 25 is a simplified block diagram of the μFFT-1 μBus module, incorporating a kernel FFT

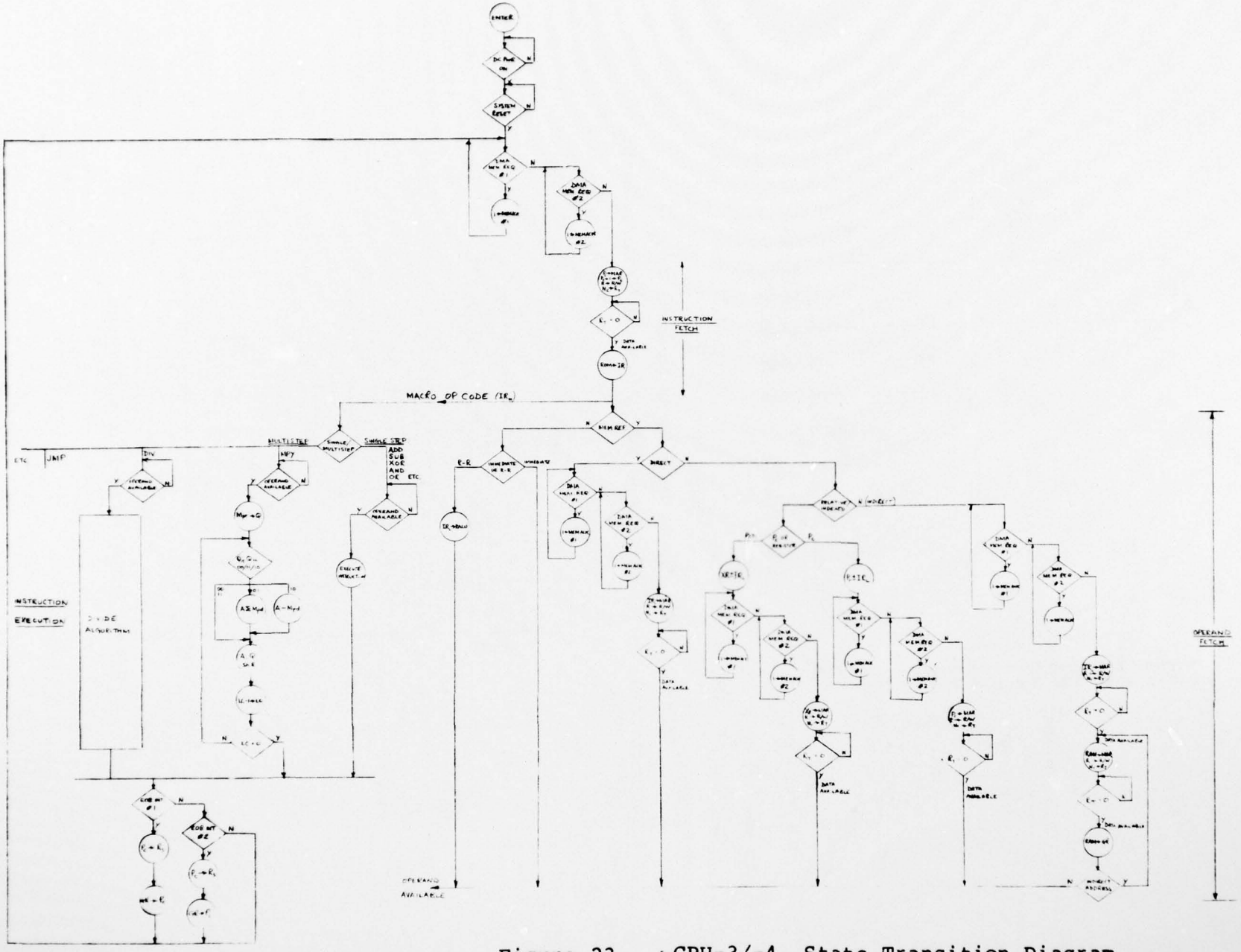


Figure 23 ΛCPU-3/-4 State Transition Diagram

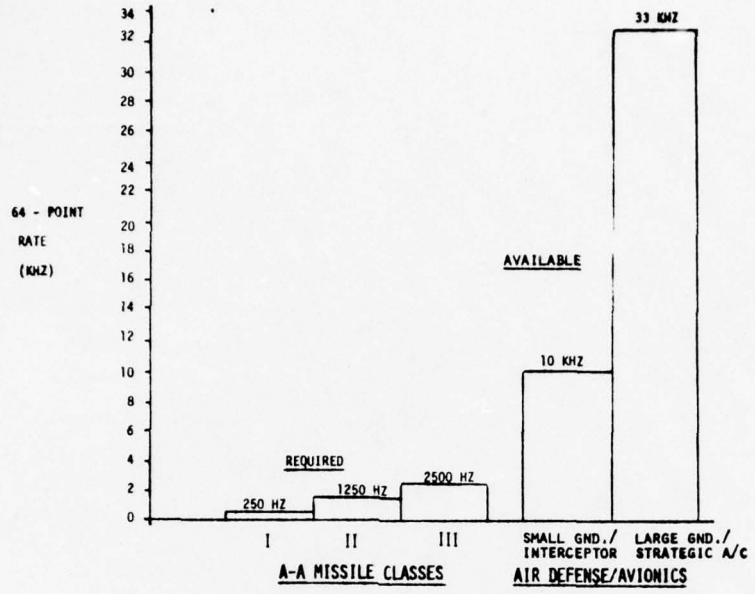


Figure 24

Missile FFT Throughput Requirements vs.
Conventional Very High Speed FFT Front-End Processors.

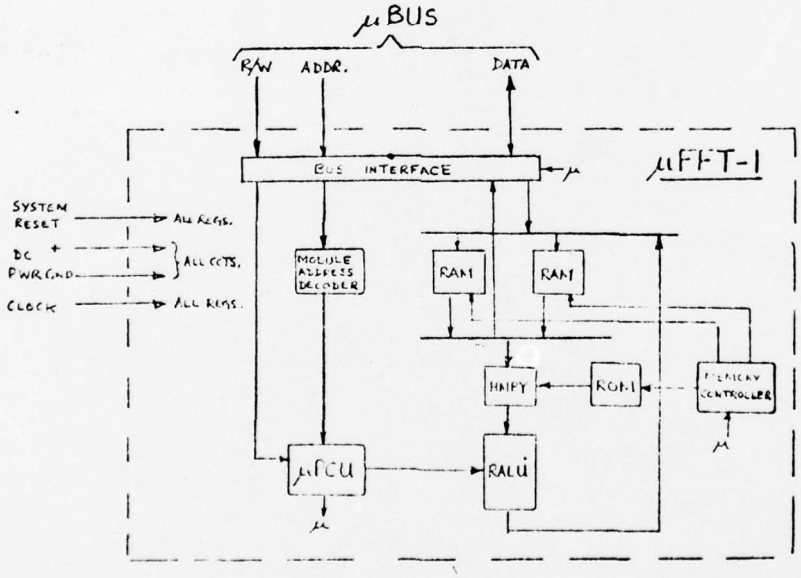


Figure 25

μFFT-1 Simplified Block Diagram

butterfly (2-point transform) arithmetic unit (μFFT), which uses the same bit-slice RALU circuit modules as the gp μCPUs e.g.. AMD 2901. A hardware multiply HMPY circuit module is also used in this μFFT μBus module.

In the Phase II Report the FFT process was presented and defined as requiring $N/2 \log_2 N$ complex arithmetic operations. This complex arithmetic operation is known as a bufferfly as represented in Figure 147 of the Phase II Report. For a 64-point transform, 192 complex arithmetic operations are required. It was shown that the butterfly operation is composed of 4 multiply 3 add and 3 subtract operations.

Investigation into the programming of the basic 2-point transform kernel (Butterfly) of the FFT algorithm is not simply the 10 steps i.e., 4 multiplies and 6 adds/subtracts but a function of the architecture, the organization of data in the memory (local store) and the address structure of the RALU. The micro instruction count for the basic 2-point transform kernel is in the order of 30 steps for an AMD 2901-based butterfly with hardware multiply, but without optimization techniques. The instruction count includes all the loads, stores, multiplies and adds that constitute the 2-point transform. For an execution time per micro instruction of 200 nanoseconds, the time to process the basic butterfly is 30 x 200 ns = 6 μsec and for the 64-point transform which requires 192 x 6 μsec = 1.152 millisec.

However, examination of the FFT reveals that the data could be subjected to a partitioned organization wherein pairs of values could be accessed simultaneously, thereby eliminating the usual instruction cycles needed for data access. This feature, in combination with the use of a multiplier element in a data path

between the data memory and the summation microprocessor, results in a significant instruction saving. The realization of a technique to achieve this computational efficiency of 8 steps to perform the fundamental butterfly process termed Parity Organized FFT processing, is described in Appendix B. (A butterfly cycle time of 8 x 200 ns or 1.6 μsec is achieved and a 64-point transform is executed in 192 x 1.6 μsec = 307.2 μsec i.e. a computation improvement of 30 steps down to 8 steps with a reduction of 67%. This RALU-based design approach achieves both low-cost circuit commonality with the other macromodules and provides an FFT execution time/meeting the requirements of Class I, II and III missiles i.e. 4.2, 0.84 and 0.41 msecs respectively.

4.2.5 HMPY-1 Module Definition

This macromodule utilizes the same hardware multiplier LSI circuit (HMPY) used in the FFT butterfly module (μFFT) described in the previous subsection, and adds to this basic multiplier chip the necessary interface circuits for μBus compatibility. The μCPU treats this module as a memory module by writing/storing two operands (a multiplier and multiplicand), performing a no-operation, and reading/loading the resulting product back into its accumulator. Timing analyses have shown this to offer a distinct throughput improvement in cases where no multiply instruction is available in the processor's instruction set and where the number of multiply operations required seriously affects microcomputer performance. In the case of the 8080A, the 8 x 16 bit multiply time is reduced from approximately 126 μ sec using a software subroutine to 17 μ sec with the HMPY-1 μBus module, a 7:1 speed improvement.

Figure 26 is a functional block diagram of the HMPY-1 macromodule assuming a low-power, CMOS-SOS equivalent of the TRW MPY-16 circuit as the kernel element.

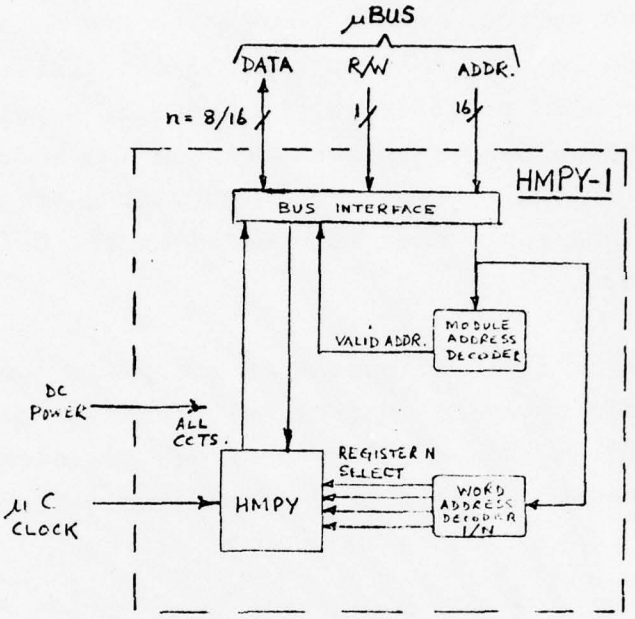


Figure 26
HMPY-1 Functional Block Diagram

The operating sequence is simply as follows:

1. Write multiplier into R_1
2. Write multiplicand into R2
3. Wait multiply time
4. Load most significant half of product (MSP) in output register
5. Read MSP

4.2.6 DMAIO Module Definition

The direct memory access input-output (DMAIO) module provides the μBus access, memory addressing and data transfer control functions for the ADAC and SDIO I/O interface macromodules. The DMAIO module also provides for high-speed, parallel digital, inter-micro computer data transfers between RAM modules. The latter mode was initially provided to support a front-end signal processor which has since been eliminated by the definition of a μBus μFFT module.

Figure 27 is a block diagram of the DMAIO module. It incorporates a microprogram control unit (μPCU), using LSI/MSI circuit modules from the same family of devices used by μCPU-2, -3 and -4, together with a register file and logic function element.

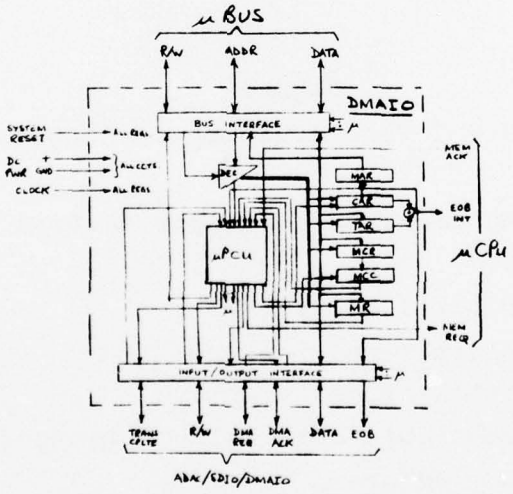


Figure 27
DMAIO Module, Block Diagram

The module interfaces with the standard microbus in the same manner as main memory modules to:

(1) accept initializing data from the μCPU under program control and

(2) to transfer data to/from main memory read/write modules (RAMs) and interfacing input-output modules, (i.e. ADAC, SDIO, DMAIO), in response to DMA requests from the latter and by memory cycle stealing from the μCPU during the execution of the operating program.

DMAIO Functional Operation - Figure 28 is a state transition diagram depicting the various functional states of the DMAIO for each operating mode, in synchronism with the μCPU clock.

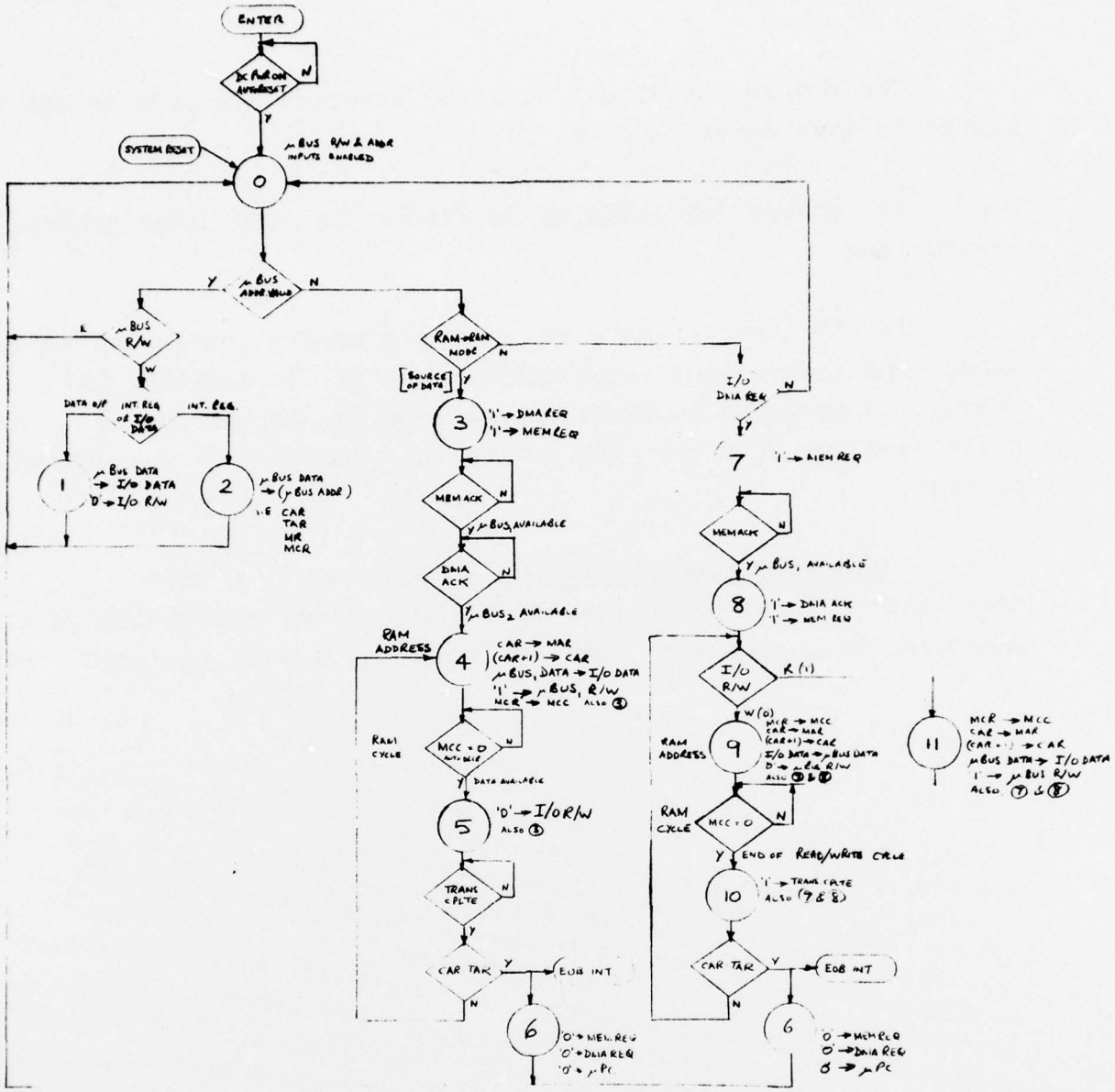


Figure 28

DMAIO Module State Transition Diagram

DMAIO operating modes total four, to accommodate:

(1) the initializing of registers within the DMAIO through the execution of "Store" type instructions by the μCPU;

(2) the initializing of interfacing I/O modules (i.e. ADAC & SDIO), in a similar manner to (1), by transferring a single control word through the DMAIO to the ADAC or SDIO;

(3) the automatic transfer of data to/from RAM and the ADAC or SDIO modules in response to transfer requests from the latter modules;

(4) the intermicrocomputer RAM to RAM data transfer, for high-speed digital data transfers between a radar signal processing microcomputer and a follow-on "post-processor" microcomputer.

DMAIO Initializing Mode

This mode is entered by the μCPU addressing the DMAIO and storing words at specific addresses in the same manner as RAM transfers. The μBUS address lines are used to point to individual registers within the DMAIO.

I/O Module Initializing Mode

This mode is similar to the previously described mode with the exception that the μCPU addresses the output port of the DMAIO module as a destination for a single word data transfer. The word transferred contains control data for:

(1) selecting the required system sampling interval and A-D convertor sampling rate;

(2) initiating automatic sampling and DMA by the A-D;

(3) initiating automatic DMA by the D-A convertor;

(4) initiating automatic DMA by the SDIO module for data outputting.

The transfer sequence is as follows:

(1) The μCPU executes a store instruction, addressing the DMAIO module and its output port.

(2) The DMAIO module executes a microinstruction enabling the I/O Data output drivers and setting the I/O R/W line to '0', i.e. write, also the DMA request line to '0'.

(3) The ADAC/SDIO stores the command word in the control register.

I/O Module Initiated Data Transfer to RAM

Following the initializing of ADAC or SDIO modules by appropriate μCPU "store" instructions, data transfers between RAM and the respective I/O module are initiated by the module through a DMA request to the DMAIO module. A-D and D-A direct memory access operation are mutually exclusive. The operating sequence is as follows:

(1) The I/O module (ADAC/SDIO) enables the DMA REQ line to the DMAIO module.

(2) The DMAIO module enables its MEM REQ line to the μCPU.

(3) The μCPU responds by raising the MEM ACK line to the DMAIO.

(4) The DMAIO module in turn raises the DMA ACK line to the I/O module to indicate the availability of the μBUS for RAM transfers.

(5) The I/O module requests either read/write via the R/W line. The I/O module places data on the data input lines to the DMAIO for writing data into RAM.

(6) For writing data into RAM, the DMAIO module transfers the contents of the current address register to the memory address register MAR; increments, CAR; connects the MAR register and input data lines to the μBUS; sets R/W to write ('0'); loads the memory cycle counter with the number of clock pulses required for the RAM write interval stored in the memory cycle register (MCR) and commences the count-down of MCC, each subsequent clock cycle.

(7) When MCC = 0 the DMAIO module raises the TRANS CPLTE line to the ADAC module.

(8) The sequence repeats itself using the R/W line to the DMAIO as the means of requesting further transfers to RAM. The MEM REQ to the μCPU is maintained high until all data transfers have been made.

(9) When the contents of CAR = TAR the end-of-block interrupt line is raised, the MEM REQ line disabled and the micro program counter (μPC) cleared.

(10) The EOB interrupt line is subsequently disabled by the μCPU re-initializing the CAR register.

The sequence is similar for reading from memory to the D-A convertor except that data is transferred from the DMAIO module to the D-A convertor holding register when TRANS CPLTE is enabled.

DMAIO to DMAIO Mode

This mode is used for intermicrocomputer RAM to RAM high-speed data transfers. Data is transferred from RAM to RAM sequentially after the Busses have been secured and the μCPUs locked-out until the block transfer is completed.

The operating sequence is as follows:

(1) $DMAIO_1$ is initialized by μCPU_1 i.e. CAR, TAR, MCR and mode register (MR)

(2) RAM RAM mode executed.

a) μBUS_1 accessed through request to μCPU_1

b) μBUS_2 accessed by DMA request to: interfacing $DMAIO_2$ and then to its μCPU_1.

c) with μBUSSES obtained $DMAIO_1$ initiates a RAM read cycle.

d) When read cycle is complete and data is stable $DMAIO_1$ requests a write cycle to DMAIO2

e) $DMAIO_2$ initiates a write cycle to RAM_2. $DMAIO_1$ waits for trans. cplte.

f) When write cycle is complete $DMAIO_2$ indicates this to $DMAIO_1$ via "Trans. Cplte" line.

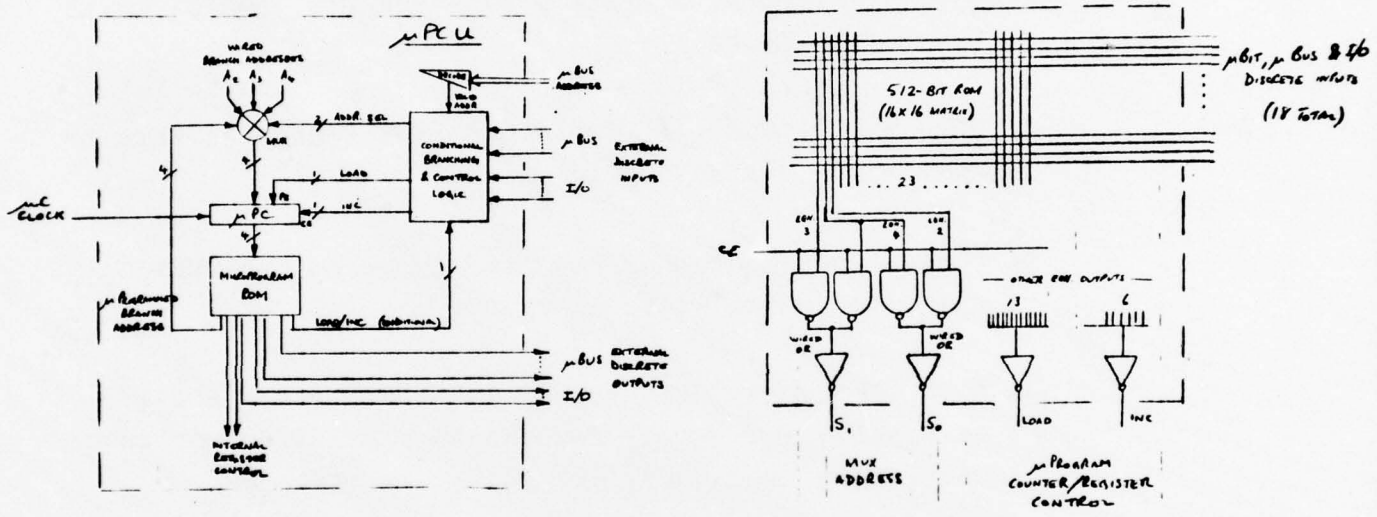
g) $DMAIO_1$ then addresses new RAM, location, and $DMAIO_2$ waits for next "write" command.

h) When $DMAIO_1$ has transferred last data word, EOB int. is enabled and $DMAIO_1$ returns to state 1 by no longer requesting the μBUS_1 and μBUS_2 via 'DMA req'.

Similarly, $DMAIO_2$ relays the μBUS 'let go' in response to the DMA req. being disabled.

μPCU - Figure 29 (A) and (B) provide more detailed block diagrams of the μPCU and the conditional branching and control logic respectively. The latter being more economically implemented with a programmable logic array (PLA).

RALU-based μPCU - The register file and logical functions provided by the RALU LSI circuit (i.e. AMD2901) offer a convenient means of implementing the DMAIO module.



(A) μ PCU

(B) Conditional Branching and Control Logic

Figure 29
DMAIO Microprogram Control Unit (μ PCU).

Figure 30 illustrates a general scheme for this form of hardware mechanization.

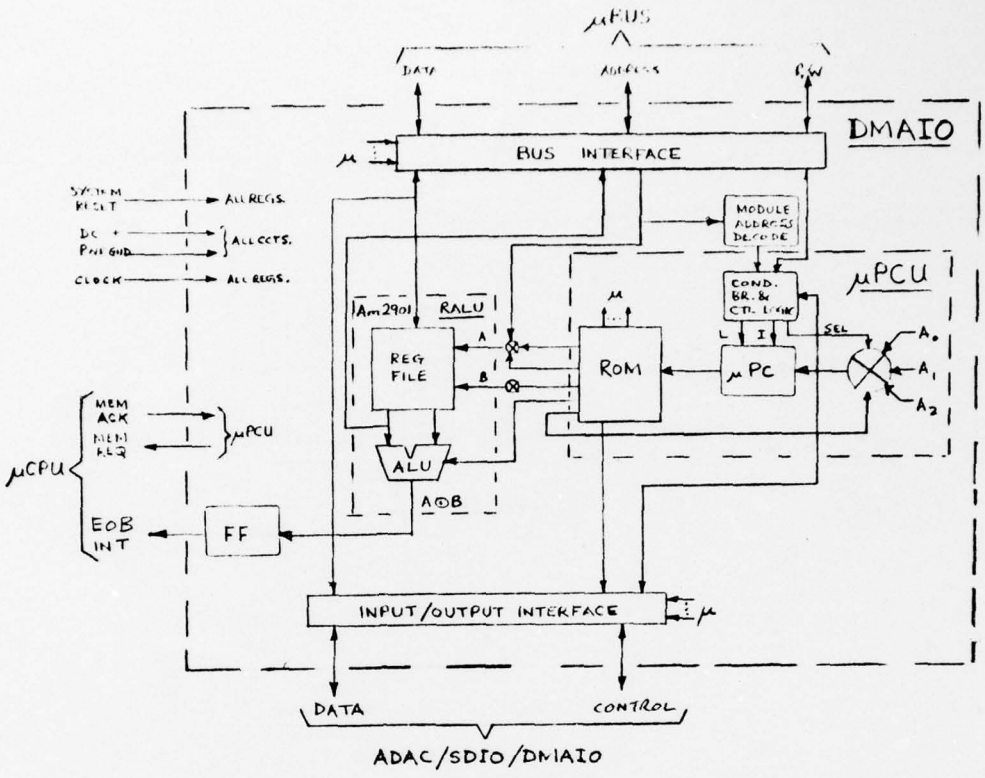


Figure 30

RALU-Based DMAIO Module, Block Diagram

4.2.7 ADAC Module Definition

A single composite analog-to-digital and digital-to-analog convertor module, (ADAC) fulfills the data conversion requirements of missile G&C systems. Figure 31 is a block diagram of the ADAC module. This module is also microprogram controlled to enable different operating modes and sampling sequences to be selected through initializing commands transferred by the μCPU via the μBus and DMAIO module.

AD-A042 466 RAYTHEON CO BEDFORD MASS MISSILE SYSTEMS DIV F/G 16/4.1
MODULAR DIGITAL MISSILE GUIDANCE PHASE III.(U)
MAY 77 F J LANGLEY
UNCLASSIFIED BR-9448 ONR-CR233-052-3 N00014-75-C-0549 NL

2 OF 3
AD
A042466

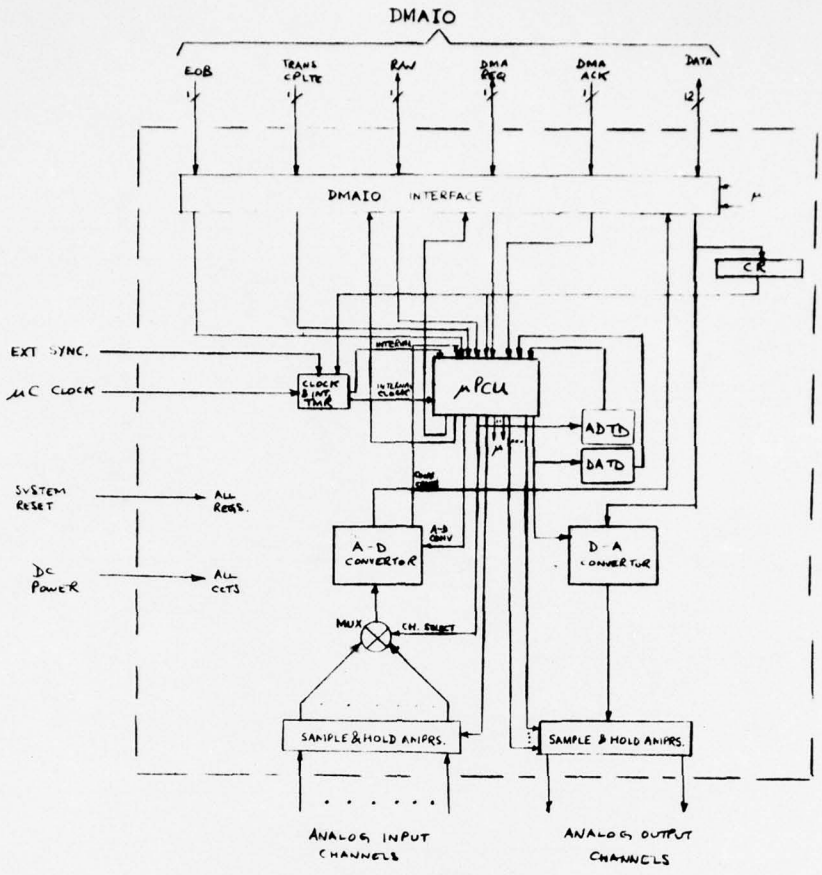


Figure 31
ADAC Module Block Diagram

System Requirements - To cover the range of analog interface requirements for Class I, II and III missiles, (see Table 9), provision is made for the simultaneous sampling of 8, 16 or 24 analog input channels synchronized to an external reference for either rate gyro carrier frequency demodulation, (i.e. peak sampling), or pulse radar range gating. A-D conversion time and degree of digital quantization can be preset for short or long conversion cycles, thereby using the same A-D convertor to provide 3, 6 or 8 μsec conversions for 8, 10 or 12-bit quantization levels respectively. This covers the needs of 8-bit high-speed radar video conversions and the lower-speed higher precision 10 and 12-bit body motion rate gyro and accelerometer input quantization.

TABLE 9

ADAC SYSTEM REQUIREMENTS

SYSTEM INTERFACE	NO. OF CHS. I/P	NO. OF CHS. O/P	LEVEL H/L	POL-ARITY U/B	MOD/ UNMOD	SAMPLING RATE/CH. (Hz) I	II	III	SYNCHRONIZING IT/GR/WFG/PRG I	II	III	NO. OF BITS I	II	III	CONVERSION TIME/CH (μ secs) I A-D	I D-A	II A-D	II D-A	III A-D	III D-A
Radar Acq.	3-21	1-2	H	B	Unmod	12,800	12,800	12,800	IT	WFG	WFG	8	8	8	26	5	7	5	3.7	5
Receiver Track	7-9	1-2	H	B	Unmod	12,800	12,800	12,800	IT	WFG	WFG	8	8	8	11	5	8.6	5	8.6	5
Gimballed Platform																				
Gyros:	2-3	-	H	B	Mod.	250	500	500	IT/GR	IT/GR	IT/GR	10	12	12	*6	-	*8	-	*8	-
Potrs:	2	-	H	B	Unmod.	250	500	500	IT	IT	IT	10	12	12	*6	-	*8	-	*8	-
Torquers:	-	2-3	H	B	Unmod.	250	500	500	PRG	PRG	PRG	10	12	12	-	5	-	5	-	5
Autpilot																				
Gyros:	2-3	-	H	B	Mod.	250	500	500	IT/GR	IT/GR	IT/GR	10	12	12	*6	-	*8	-	*8	-
Accelrs:	2-3	-	H	B	Unmod.	125	250	250	IT	IT	IT	10	12	12	*6	-	*8	-	*8	-
Fin Actos:	-	2-4	H	B	Unmod.	250	500	500	PRG	PRG	PRG	10	12	12	-	5	-	5	-	5

LEGEND:

I/P - Input
O/P - Output
H/L - High or low level
U/B - Unipolar or bipolar

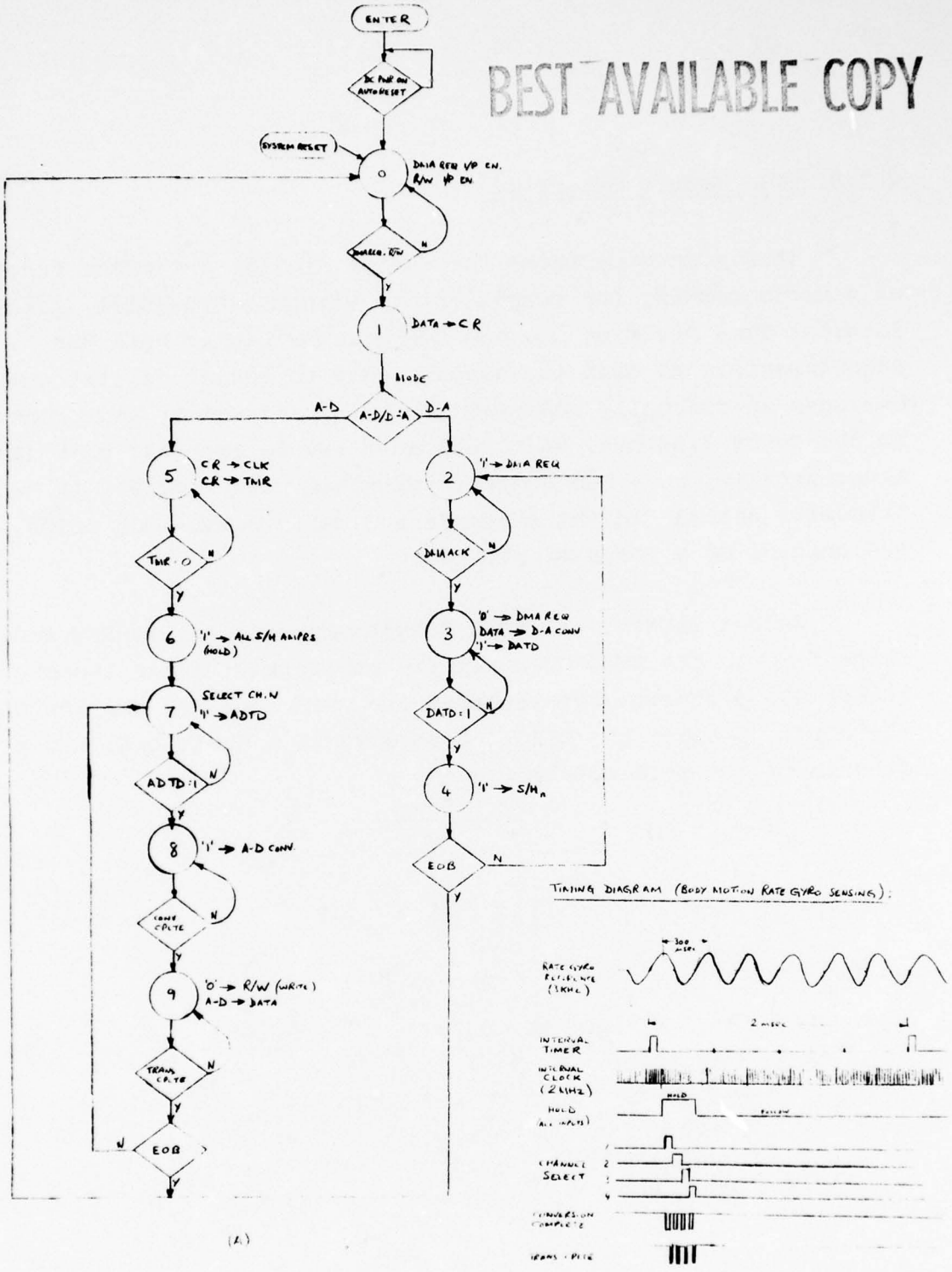
MOD/UNMOD - Modulated or unmodulated
IT - ADAC interval timer
GR - Gyro reference frequency
WFG - Radar waveform generator
PRG - Program initiated

* - Conversion time based on total allowable computational delay.

A built-in, programmable interval timer enables the automatic initiation of body motion sensor or CW radar video channel sampling at pre-programmed time intervals, in order to perform A-D conversions and DMA transfers to a μBus RAM module without burdening the μCPU and software with this repetitive I/O function.

D-A conversion and distribution of parallel digital multiplexed data from a μBus RAM module to gimballed platform torquers or fin control actuators is provided by a single 12-bit, 5 μsec D-A convertor and 8 separate sample and hold amplifier-drivers. While no more than 3 axes or 4 fins are employed for the torquer/fin interfaces, the additional outputs are used for end-around loop testing of the ADAC/DMAIO modules in conjunction with the excess analog input channels.

ADAC Functional Operation - The ADAC module has two modes of operation i.e. A-D and D-A, governed by the μCPU initializing command. This command word can be programmed to: select the interval timer; the time interval between sampling sequences for the cyclic sampling of the analog inputs; the frequency source for A-D sampling (μC clock/gyro reference/radar waveform generator strobe); and the channel sampling sequence (channel numbers/sequential/random). Figure 32 (A) shows the state transition diagram of the ADAC module and Figure 32 (B) is a timing diagram showing the internal waveforms for the simulataneous sampling of analog inputs synchronized to the peak amplitude of a 3KHz rate gyro reference frequency.



NOTES

1. APPLICABLE COMPONENT DELAYS:
   a) D-A CONVERTORS (DATD): 5 μsec
   b) SAMPLE & HOLD (S/H) AMPLIFIERS 50 nsec
   c) A-D CONVERTORS: 10/12-BIT: 6/8 μsec (BODY MOTION SENSORS)
      8-BIT: 3 μsec (RADAR RECVR INTF.)
   d) ANALOG MULTIPLEXER (MUX): 0.5 μsec

Figure 32

ADAC State Transition Diagram and A-D Sampling and Conversion Waveforms.

4.2.8 SDIO Module Definition

This module performs the serial digital interface functions of a microcomputer for compatibility with MIL-STD-1553A, Aircraft Internal Time Division Command/Response Multiplex Data Bus requirements. As such it responds only to serial digital command messages specifically addressed to it as a terminal unit connected to the party line bus, when used as a remote terminal unit (RTU). Alternatively, as a bus control interface unit, (BCIU), it transmits serial digital commands and data to RTU-mode SDIOs, under the control of a computer program.

Before launch, all SDIO modules operate in the RTU mode, responding to the AWCS BCIU via the umbilical. After launch, the seeker/SCG microcomputer replaces the AWCS computer and the SDIO of the former becomes the BCIU. Figure 33 is a functional block diagram of the SDIO module.

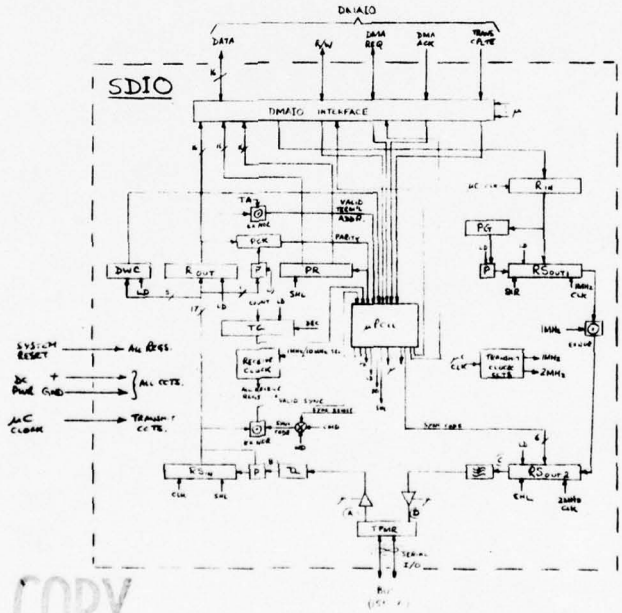


Figure 33
SDIO Module, Block Diagram

The SDIO is fundamentally a microprogram-controlled, serial to parallel, parallel to serial digital convertor module interfacing with the DMAIO (parallel digital) and missile inter-subsystem 2-wire bus (serial digital). It has two mutually exclusive operating modes: receive and transmit, or half-duplex in telecommunications terminology. A common input/output transformer couples both receive and transmit circuits to the external bus.

Serial Digital Codes & Message Formats - The serial digital pulse code modulated (PCM) code used is Manchester II, (also known as bi-phase level (Bi Ø-L) and split phase), transmitted and received at a 1 Mb/sec rate within the tolerances specified in Ref R-10.

Three different word formats are used, viz: command, data and status, (see Figure 34), each 20 bit intervals long and containing a synchronizing code of three bit periods duration (MSBs) with a level transition in the middle of the second bit period, (i.e. a non-valid Manchester II code), and a parity bit covering the remaining 16-bits of information. Data word sync is the complement of the command and status word sync codes.

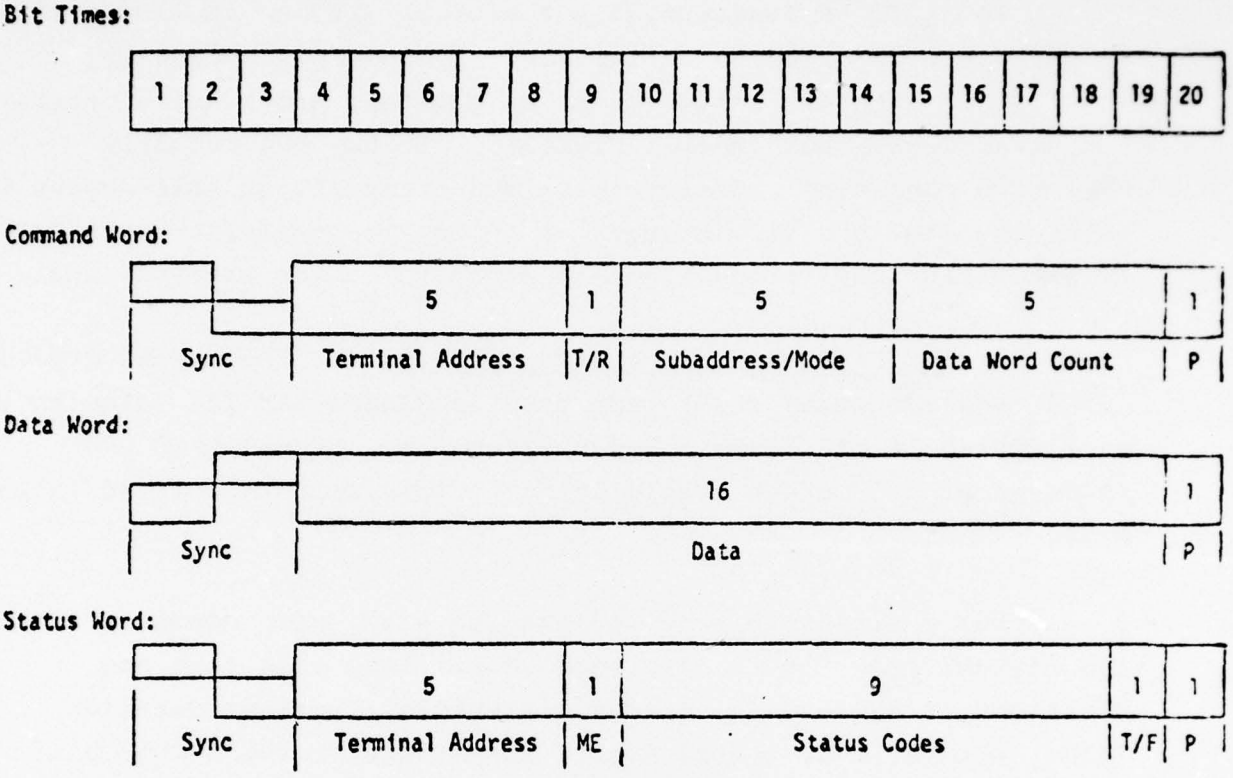


Figure 34
SDIO/1553A Word Formats

To implement the command/response system operation, three message formats are used as shown in Figure 35. These allow the master computer in the system to command a specific subsystem computer to send or receive data to/from the master computer and, at the same time, to check, (via Status words) that the addressed subsystem's SDIO/DMAIO and supporting μCPU and memory modules are functioning properly.

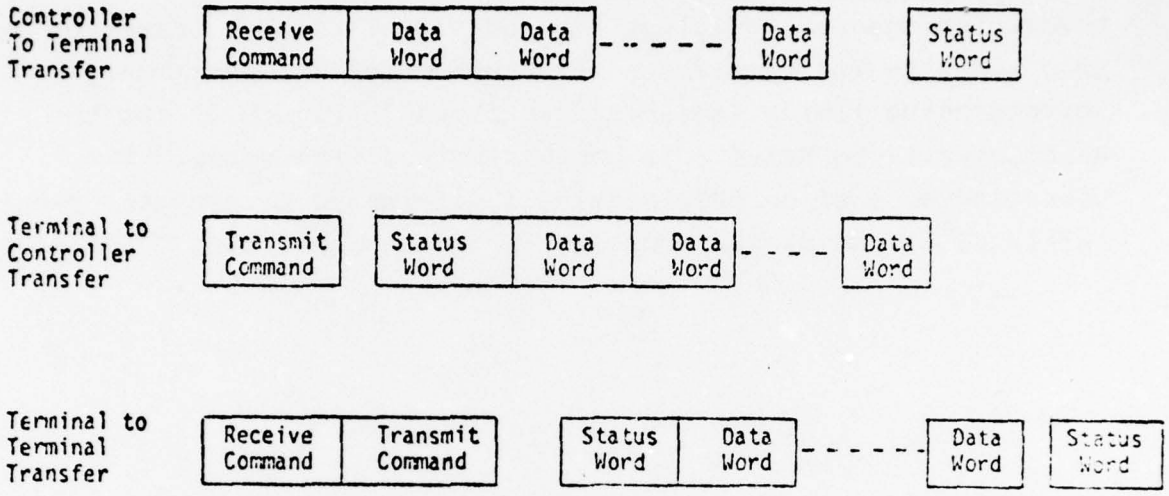


Figure 35
SDIO/1553A Message Formats

For inter slave subsystem communication the master computer can designate one subsystem to receive data (receive command) and a second subsystem to transmit data (transmit command), where upon data is transmitted from one slave computer to another.

Provision is made in the command word field assignments to provide a memory address/displacement (Subaddress/Mode) and number of words in a block (Data Word Count) to be used in a data transfer operation.

The above discussion attempts to highlight the 1553A characteristics pertinent to a missile guidance and control system. Ref. R-10 should be referred to for a more lengthy description.

SDIO Functional Operation - Figure 36 is a state transition diagram depicting the individual clocked states of the SDIO for both the receive and transmit modes of operation. Corresponding timing waveforms are shown in Figure 37 for the Manchester II to NRZ-L code conversion and vice-versa. The operating mode of an SDIO module is determined by computer program, initiated by the master computer in the case of slave SDIOs/RTUs.

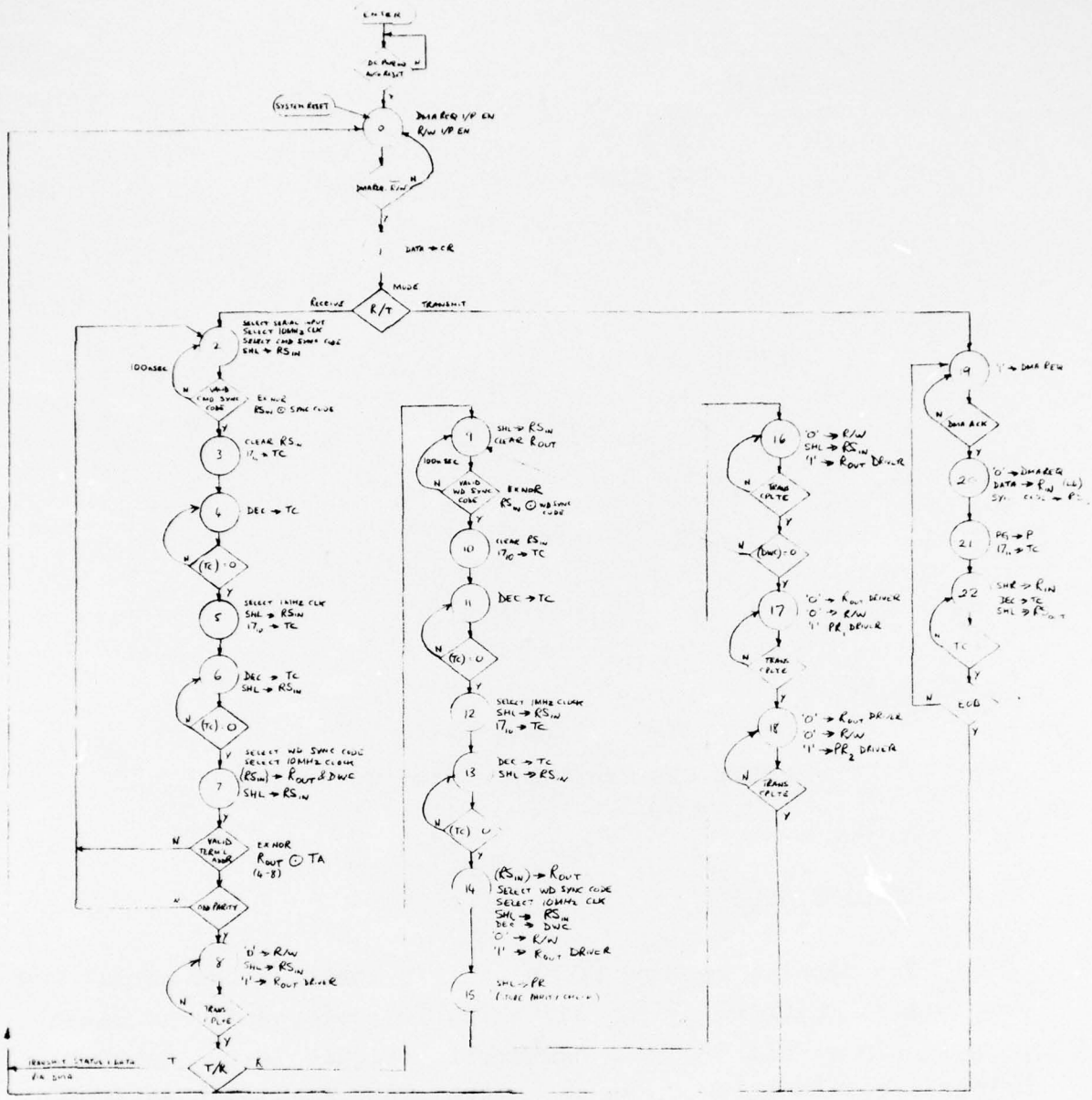


Figure 36

SDIO State Transition Diagram

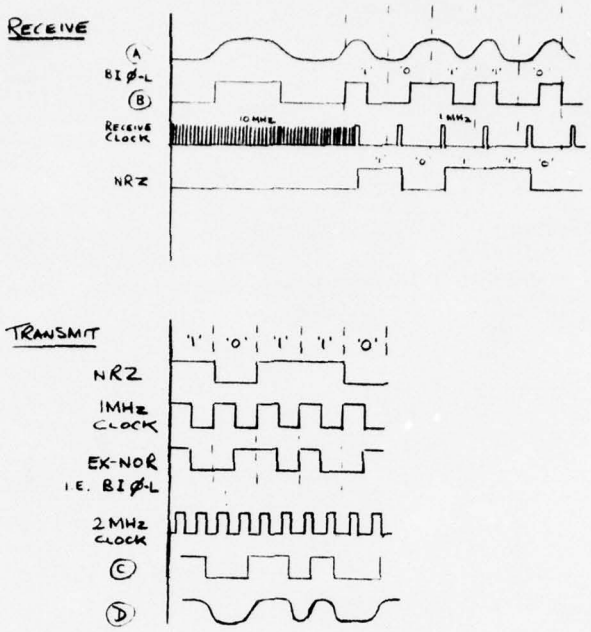


Figure 37
SDIO PCM Code Conversion Waveforms

Receive Mode

The receive command bus signal is the filtered serial PCM waveform A, (Figures 34 and 37) which is restored to the clean pulse train waveform B by a hysteresis circuit (e.g. Schmitt trigger). This pulse train is shifted into the input shift register RS_{in} at a clock rate sufficient to resolve the bit transition interval for the purposes of recognizing the command word sync. code, e.g. 10 MHz. Microinstructions in the μPCU control program select the latter clock frequency and the equivalent parallel bit pattern corresponding to the command word

sync. code at the shift rate used for comparing the contents of RS_{in} and the valid code. The occurrence of a valid sync. code results in a microprogrammed change in clock frequency such that data entering RS_{in} is shifted at a 1 MHz rate during the first 1/2 bit period of the input pulse train. Code conversion to NRZ-L is then automatic since the first 1/2 bit period of a bi-ϕL waveform determines the data bit value, i.e. 1/0. After 17 shifts, (determined by the timing counter (TC) contents being decremented to zero), the contents of RS_{in} and the 1-bit P register are parallel loaded into R_{out} the output register and parity flip-flop (P), respectively. The contents of the data word count field are also loaded into the DWC register. The latter determines the number of words to be loaded into R_{out} for any given input message. Shifting at the 1 MHz rate continues for RS_{in}. The terminal address field bits in R_{out} are compared with the valid terminal address code of the SDIO. All 16 information bits in R_{out} are checked for odd parity by the PCK circuit and the result shifted as one bit into the parity register (PR).

With a valid address and correct parity established, the R/W line to the DMAIO module is set to '0' and, after Bus access, the contents of R_{out} are transferred to a RAM module location determined by the DMAIO CA register.

The shifting of R_{in}, checking of word sync., loading of R_{out}, checking of word parity and data word transfers to RAM continues until the contents of the DWC register are zero, where upon the contents of the message parity registers are transferred to RAM via the DMAIO.

Transmit Mode

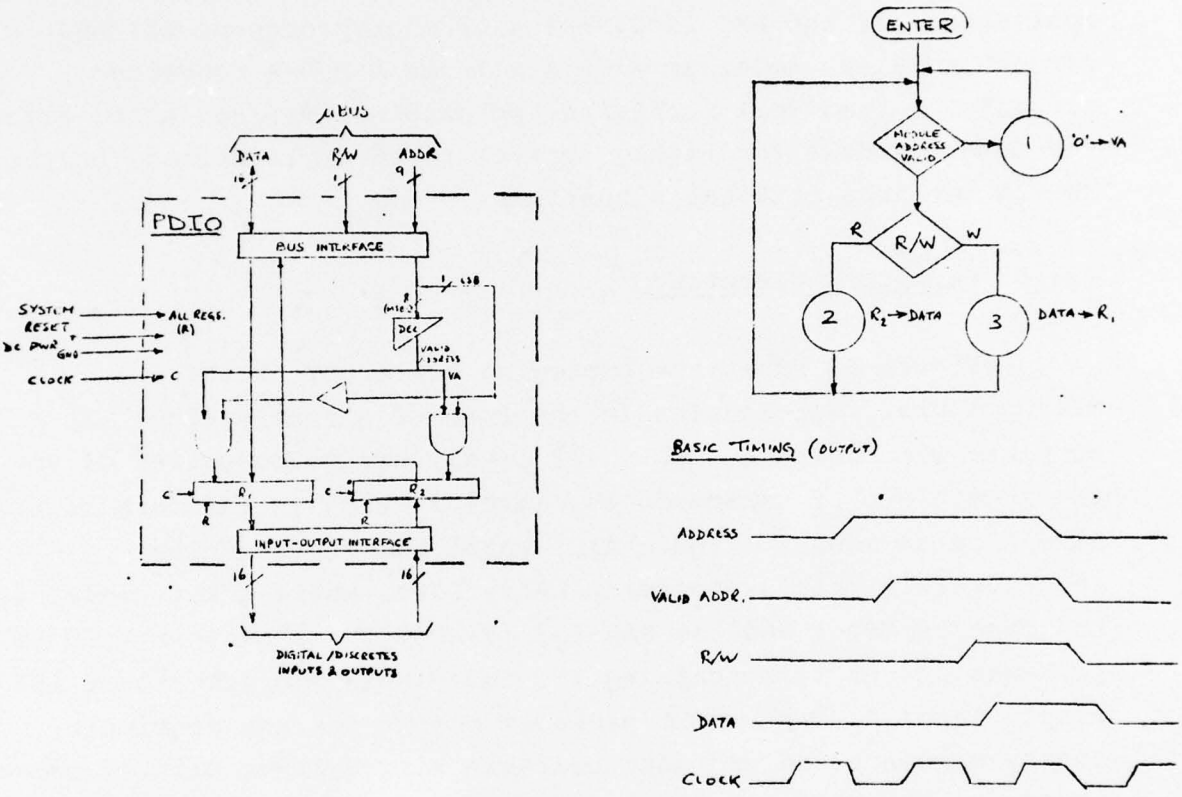
Following a receive command and data word transfer to RAM, the μCPU performs a message validation check and generates a status word for transmittal to the master computer BCIU/SDIO. The SDIO is initialized for the transmit mode of operation. The status word is parallel loaded into the R_{in} register and odd parity is generated and stored in P and the contents of Rin are parallel loaded into RS_{out1}. A 1 MHz clock is used to shift out the contents of RS_{out1} to a serial exclusive-NOR circuit where each bit of the status word is compared with the 1 MHz clock and the output shifted into the RS_{out2} shift register at a 2 MHz clock rate. A 6-bit status word sync code is loaded into RS_{in} in the MSB positions before 2 MHz shifting begins. Waveform C appears at the output stage of RS_{out2} which is the complete Manchester II coded status word. This serial pulse train is passed through a low pass filter to remove the high frequency harmonics, and waveform D is output to the inter-subsystem bus via the output transformer.

Transmit commands from the master computer are handled in the same way as receive commands, except that a status word and the required data words are output from RAM within 2 to 5 μsecs after receipt of the last bit of the command word.

4.2.9 PDIO Module Definition

This is virtually a two-word memory with one location read-only and the other write-only, to satisfy the digital discrete interfaces of a missile G&C system. Figure 38 (A) illustrates the functional layout of the PDIO, and Figure 38 (B) shows the

corresponding state transition and timing diagrams. The μ CPU addresses this module as it would a RAM to transfer the status of discrete system input lines as a whole 16-bit (or 8-bit for μ CPU-1&-2) word to a μCPU register file location. Similary, to control solenoids/relays, a word can be written into the output register of the PDIO.



(A) Functional Block Diagram

(B) State Transition and Timing Diagrams

Figure 38
PDIO Module

4.3 VLSI SEM Packaging

The Navy SEM packaging system was reviewed through discussions with NAFI, and, based upon low-power CMOS/SOS equivalents of the AMD 2900-Series of microprocessor LSI/MSI circuit modules, together with available A-D/D-A convertor circuits, a practical packaging approach was defined using either the SEM-1A module for highly form-factored applications, or the SEM-2A for less critical situations.

4.3.1 Packaging Hierarchy

Figure 39 shows the packaging hierarchy of the microcomputer macromodules in the form of a family tree. A complete microcomputer (μC), 1st level, can be comprised of any required group of macromodules, level 2, each in the form of a Navy SEM-1A or 2A module (Figure 4). These SEM macromodules incorporate either standard-industry DIPs, where practicable, as in the case of μCPU-1 and the RAM and (P)ROMs, or 2" x 1" hybrid LSI packages, level 3, containing the individual microprocessor LSI/MSI chips, level 4. The latter packages enable the use of simple, easily manufactured and more reliable single-layer printed circuit boards on the SEMs.

MACROMODULAR MICROCOMPUTER FAMILY TREE

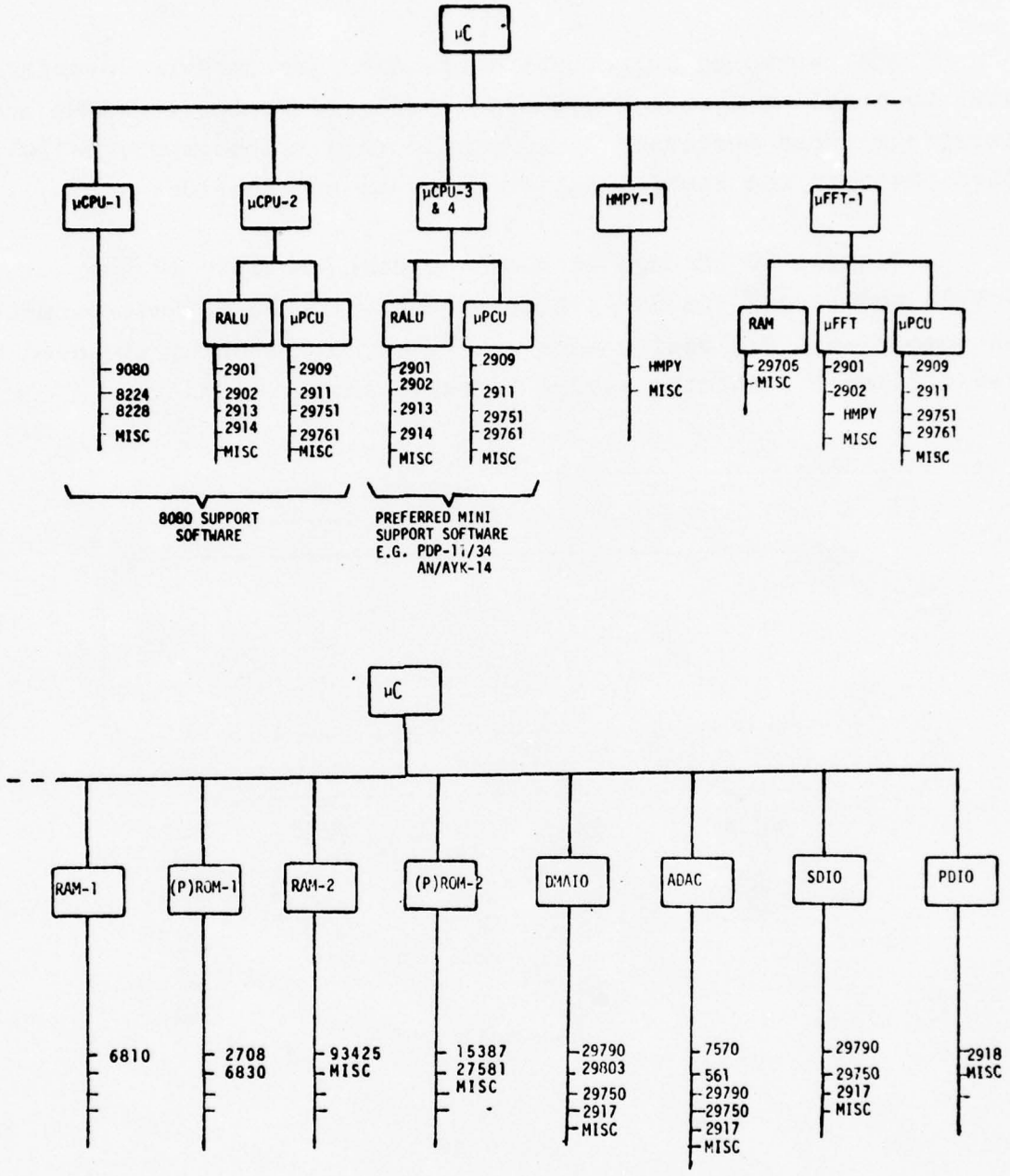


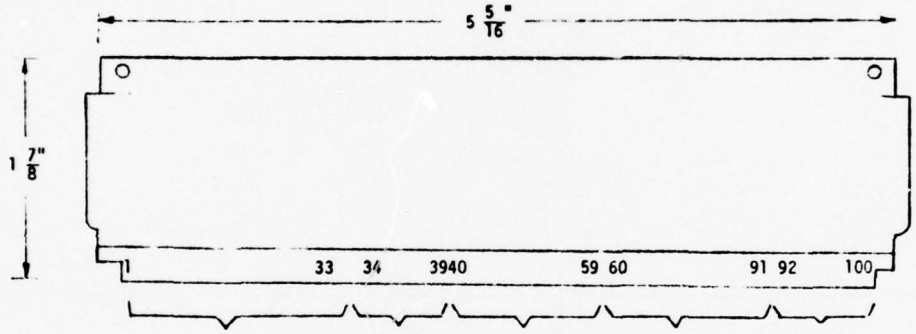
Figure 39

Macromodular Microcomputer Packaging Hierarchy

4.3.2 Standard μBus/Interface

SEM packaging supports the standard macromodule interface goal by enabling system designers to select appropriate SEMs and integrate these successfully to configure a microcomputer which in turn, matches the requirements of a given application.

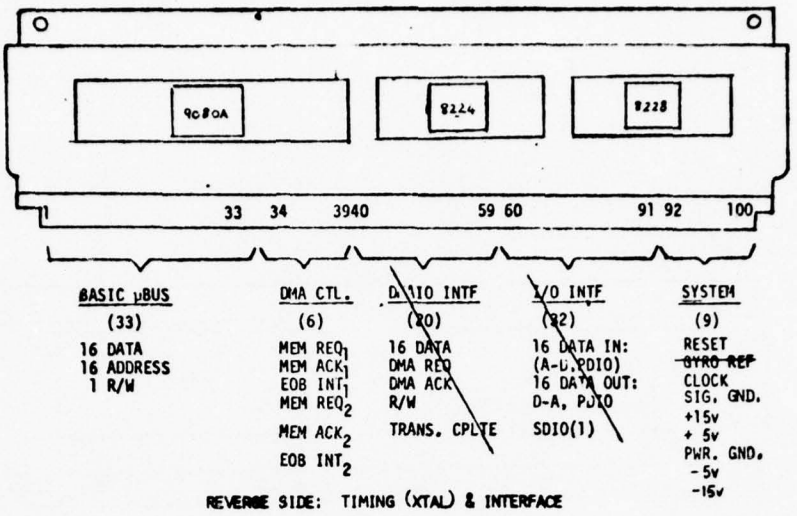
Figures 40 through 46 show the compatibility of the SEM-2A module and the 100-pin connector with the ONR microcomputer macromodules. Pin assignments are fixed, eliminating the need to use different connector keying for each module type.



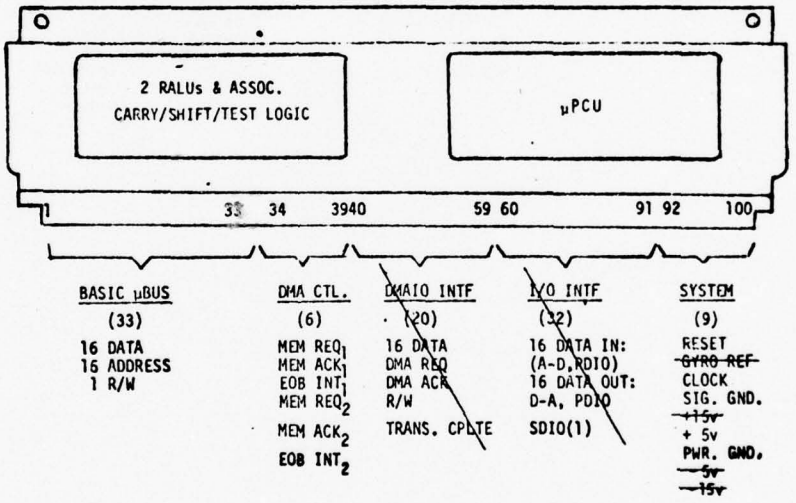
BASIC μBUS	DMA CTL.	* DMAIO INTF	I/O INTF	SYSTEM
(33)	(6)	(20)	(32)	(9)
16 DATA	$MEM\ REQ_1$	16 DATA	16 DATA IN:	RESET
16 ADDRESS	$MEM\ ACK_1$	DMA REQ	(A-D, PDIO)	GYRO REF
1 R/W	$EOB\ INT_1$	DMA ACK	16 DATA OUT:	CLOCK
	$MEM\ REQ_2$	R/W	D-A, PDIO	SIG. GND.
	$MEM\ ACK_2$	TRANS. CPLTE	SDIO(1)	+15v
	$EOB\ INT_2$			+ 5v
				PWR. GND.
				- 5v
				-15v

*DMAIO-DMAIO TRAISFERS

Figure 40
ONR/SEM-2A Microcomputer Macromodule,
Standard Pin Assignments



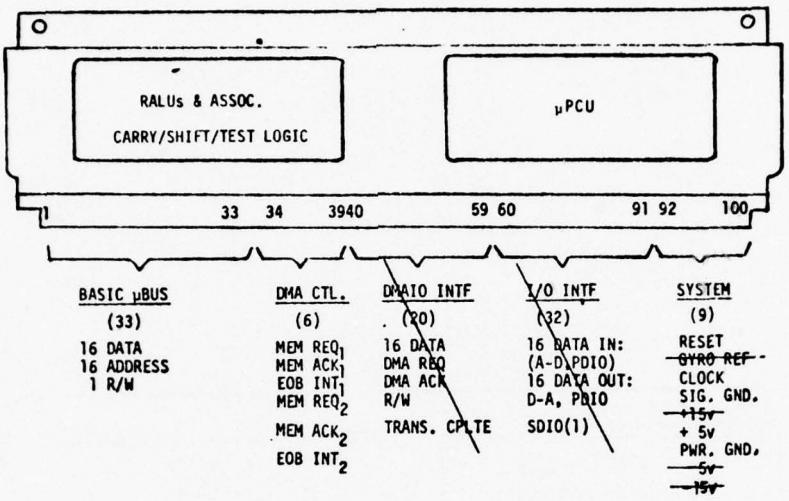
(A) µCPU-1



(B) µCPU-2

Figure 41

ONR/SEM-2A Microcomputer Macromodules, Microprocessors

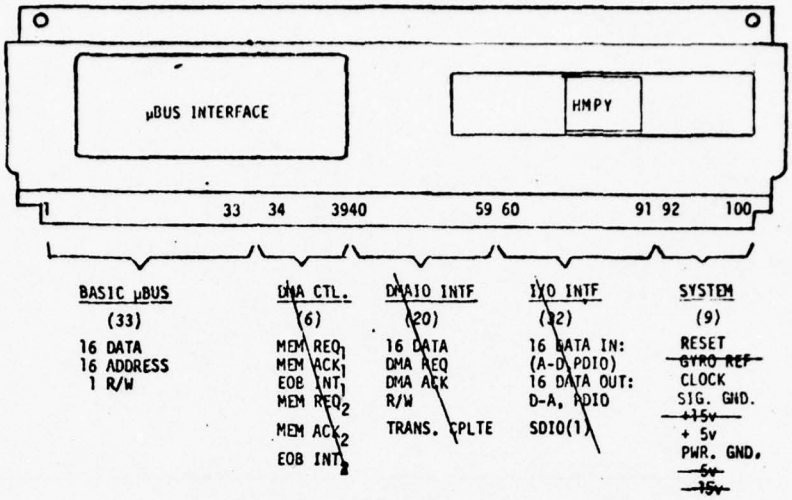


(C) μCPU-3/-4

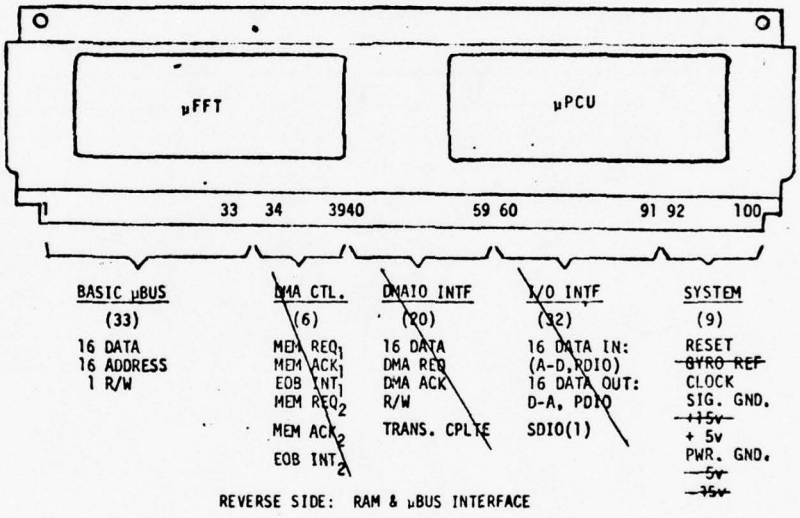
Figure 41

ONR/SEM-2A Microcomputer Macromodules,

Microprocessors (Contd.)



(A) HMPY-1



(B) µ FFT-1

Figure 42

ONR/SEM-2A Microcomputer Macromodules,
High-Speed Arithmetic

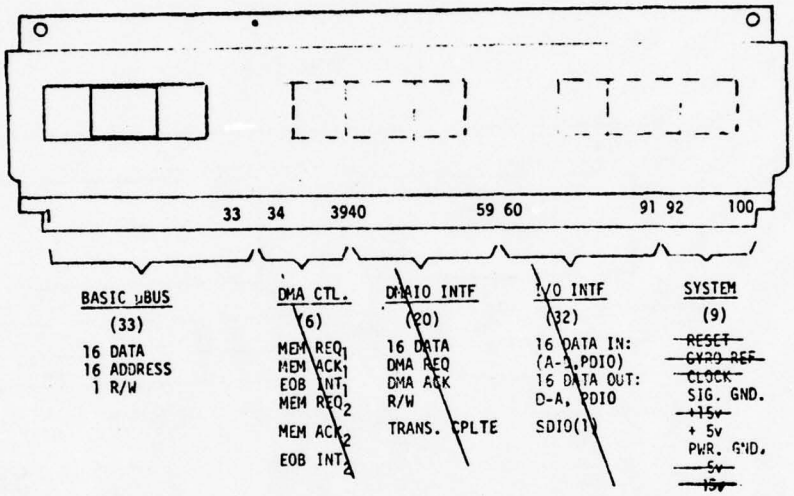
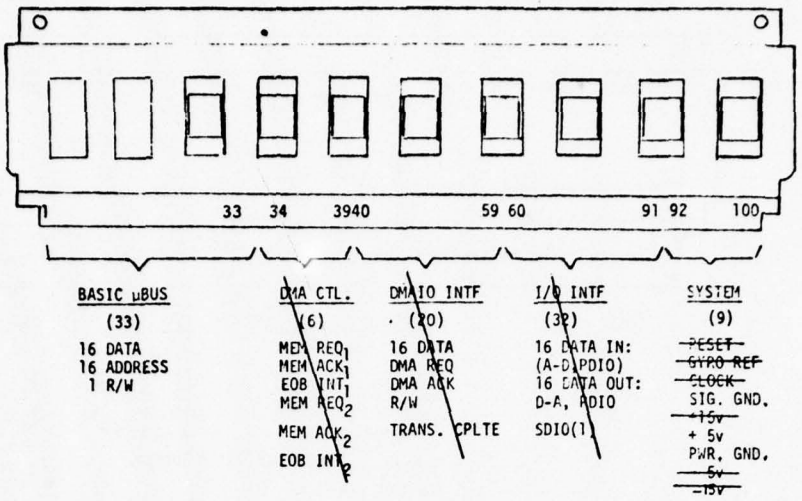
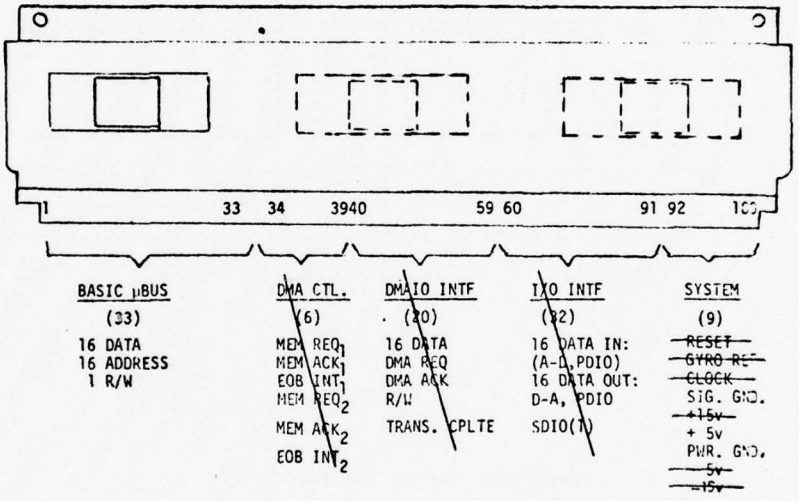


Figure 43

ONR/SEM-2A Microcomputer Macromodules,
Medium-Speed, RAM-1/ROM-1



(A) RAM-2

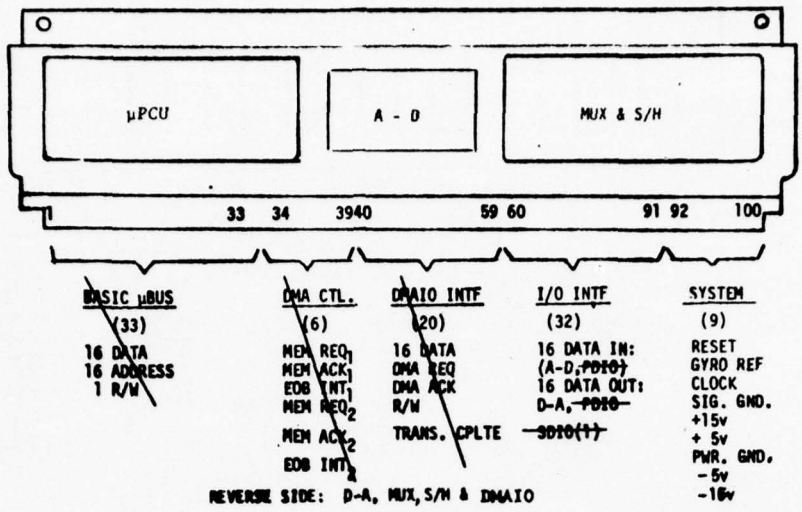


(B) ROM-2

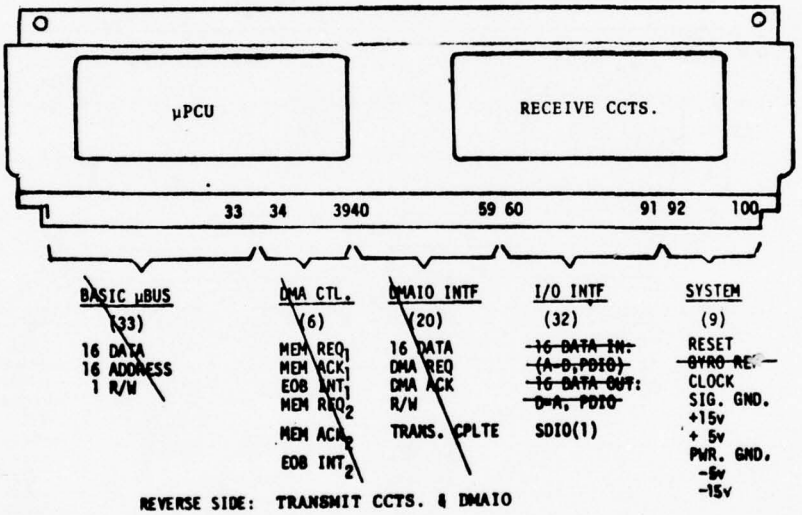
Figure 44

ONR/SEM-2A Microcomputer Macromodules,

High-Speed RAM-2/ROM-2



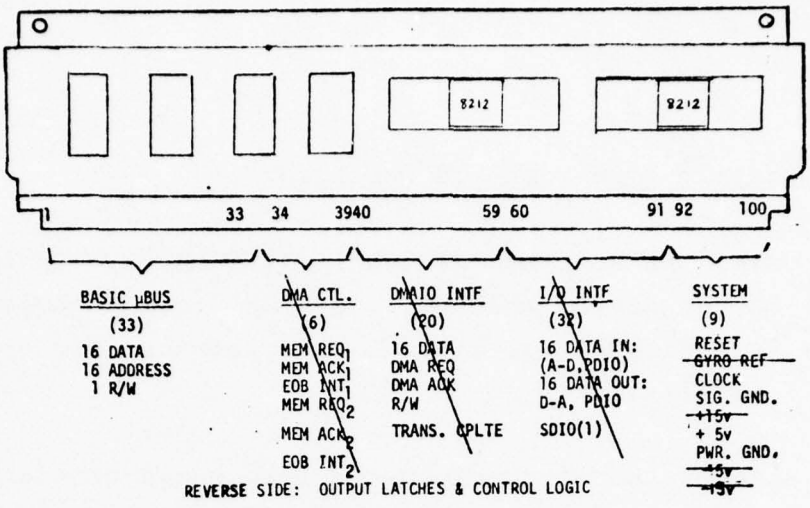
(A) DMAIO/ADAC



(B) DMAIO/SDIO

Figure 45

ONR/SEM-2A Microcomputer Macromodules,
Input-Output Interface

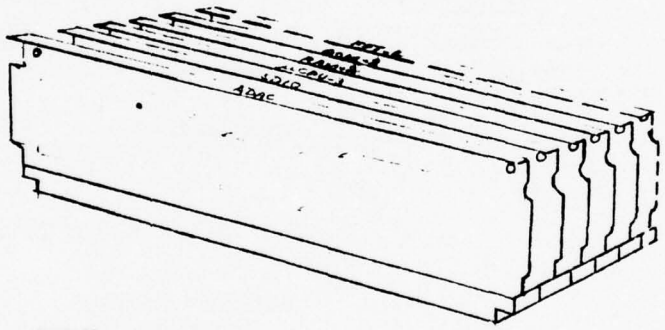


(C) PDIO

Figure 45

ONR/SEM-2A Microcomputer Macromodules,

Input-Output Interface (Contd.)



SEMs: 5ea. TYPE 2A

CIRCUIT
TECHNOLOGY: STANDARD INDUSTRY BIPOLAR /CMOS-SOS LSI, CERDIPS & HYBRIDS

MACROMODULES: μCPU-3 (AN/AYK-14 OR AN/UYK-20 LIMITED EMULATOR)

RAM-2 (256x16)
ROM-2 (1536x16)
DMAIO (2 CHS.)
ADAC 16 ANALOG INPUTS/8 ANALOG OUTPUTS
SDIO MIL-STD-1553A SERIAL DIGITAL INPUT-OUTPUT

VOLUME: 20 cu. ins. (APPROX.)

POWER: 20 WATTS (APPROX.)

FIGURE 46 ONR/SEM-2A Microcomputer

4.4 Missile Form-Factored Packaging

Using the same standard 2 in. x 1 in. hybrid-LSI package for all macrofunction modules, two VLSI, form-factored packaging schemes were defined for a 6 1/2 in. diameter, high-performance, Class II missile.

A federated microcomputer guidance and control system of the form shown in Figure 47 was used in each case, i.e. μCPU-4 based for SCG; μCPU-3 based for adaptive autopilot; and μCPU-1 based for telemetry/fuze.

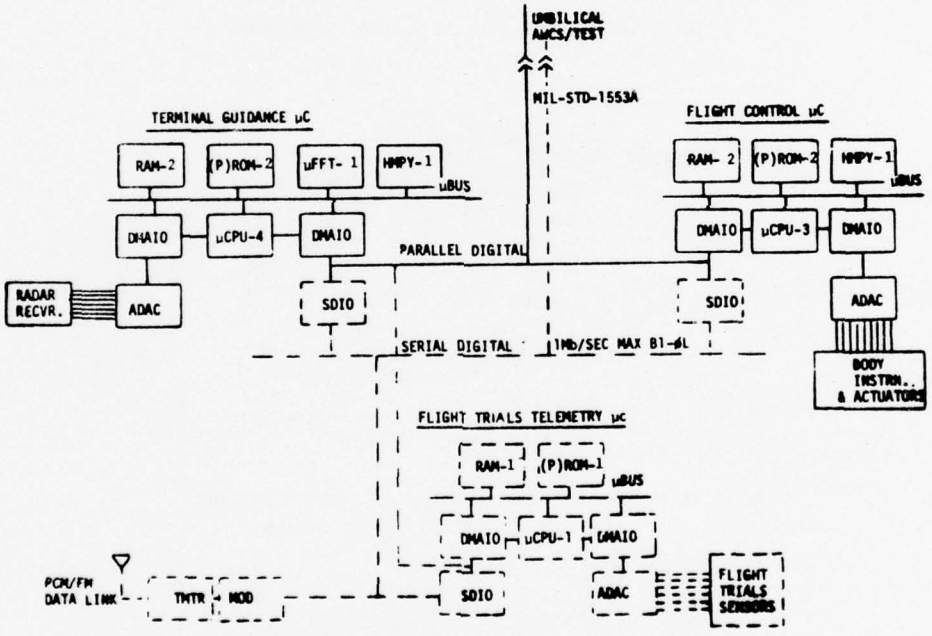


Figure 47
Federated Microcomputer System for
Form-Factored Missile Packaging

4.4.1 VLSI/SEM-1A Configuration

The three microcomputers shown in Figure 47 are shown packaged in 19 SEM-1A modules within the typical form-factored space allowance of a small 6 1/2 in. diameter missile in Figure 48 Resulting running length is only 2 in. using SEM-1A macromodules. For such a high-density packaging situation, the low-power, high-speed characteristics of CMOS/SOS are of paramount importance for the SCG and A/P microcomputers. The 8080A- based telemetry microcomputer is conveniently replaceable by the warhead fuze counterpart for final production configurations.

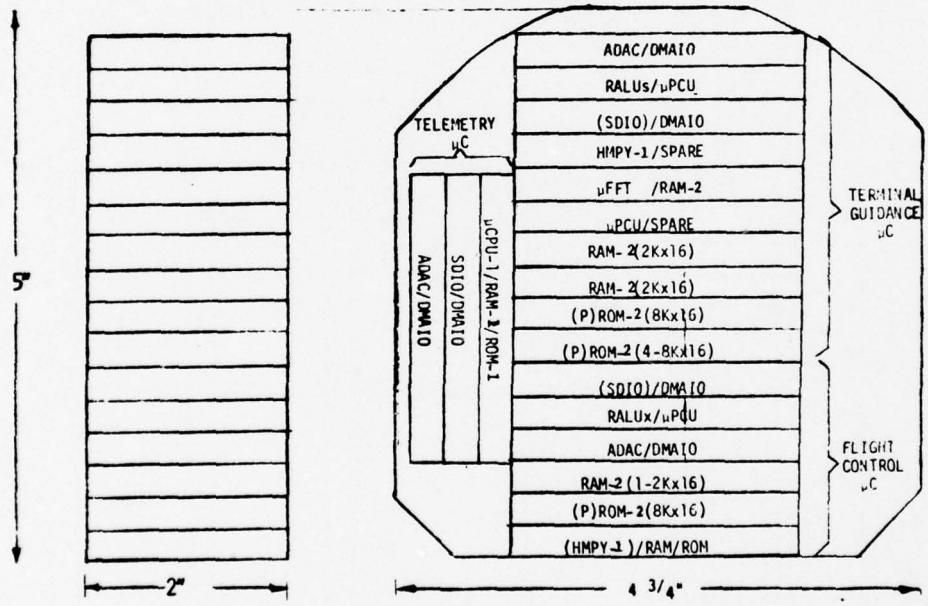


Figure 48
VLSI/SEM-1A Missile Form-Factored Packaging

4.4.2 VLSI/PCB Configuration

As an alternative to the SEM-1A packaging approach described in the previous subsection, 3 double-sided printed-circuit boards (PCBs) were configured within the same missile space and form-factor constraints. Figure 49 shows the same microcomputer macrofunction modules packaged such that: the autopilot microcomputer occupies one PCB, the SCG microcomputer one and a half PCBs and the telemetry/fuze microcomputer the remaining half of the second SCG PCB.

Both the SEM-1A and PCB packaging approaches meet the missile form-factor constraints and occupy approximately the same volume.

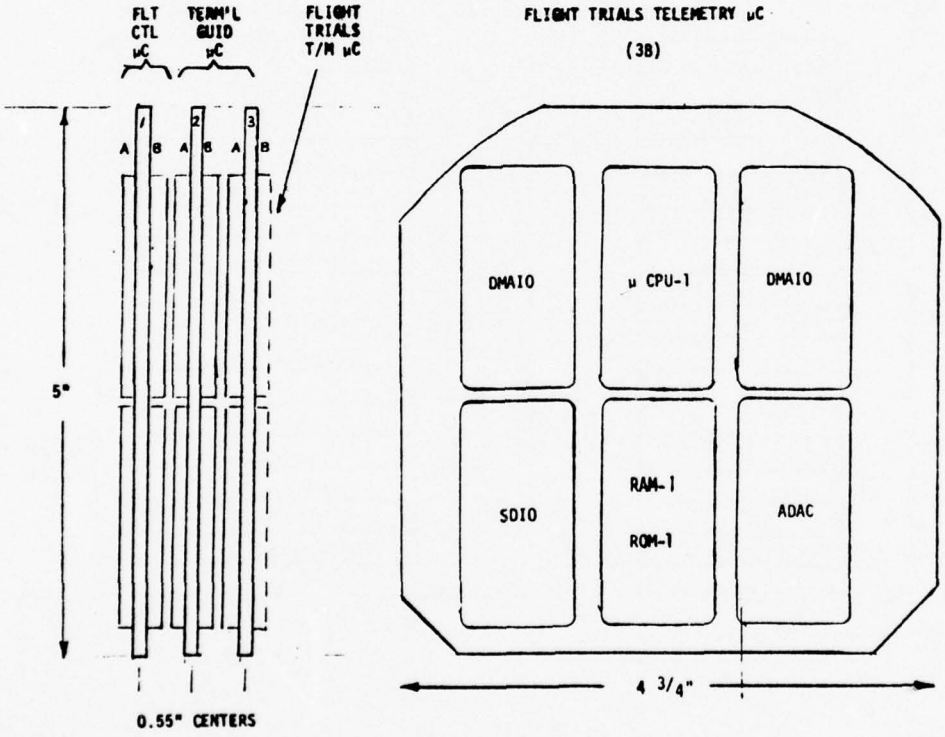


Figure 49 VLSI/PCB Missile Form-Factored Packaging

5. MODULAR MICROCOMPUTER SIMULATION

To validate the timing and control of data transfers within and between the modules previously defined (Sect. 4), module designs were simulated individually and as an interrelated group i.e. a microcomputer. The intention was to obtain system level performance data sensitive to the proposed level of modularity and internal module architecture, thereby identifying possible design deficiencies and suitable corrective measures before embarking upon a costly and relatively inflexible hardware development phase.

5.1 Simulation Methods

The choice of a suitable computer design simulation tool was based upon the following criteria:

1. Register level simulation capability

2. Part of an automated design system

3. Understandable/programmable without a hardware design background.

Register-level simulation would be adequate to validate data flow and throughput. A lower logic gate level simulation would be inappropriate, being too detailed and costly, and more applicable to integrated circuit design. Higher-level computer simulation techniques, such as those used in target machine simulators to exercise computer programs, would not provide the required functional and timing visibility in terms of I/O and register status per clock interval.

In keeping with the work performed during the previous two study phases, the simulation of modules should be part of a continuing effort culminating in hardware fabrication through the carry-over of past work without redundancy. Hence, the chosen simulation method should be related to a subsequent automated design phase for module fabrication.

Lastly, to avoid involvement/digression into unnecessary design detail, the simulation technique should be programmable by systems level users.

Taking the above factors into consideration, the following simulation methods were considered:

1. Variable architecture, micro/nanoprogrammable computer

2. Computer aided design (CAD) language for an in-house computer.

5.1.1 Variable Architecture Micro/Nanoprogrammable Computer

Two candidate machines of this type were considered: the timeshared, Programmable Research Instrument (PRIM) (Ref. R-15), which uses the TENEX timesharing system and Standard Computer Corporation MLP-900 microprogrammed computer; and the Nanodata QM-1 (Ref. R-16) a nanoprogrammed computer.

Although capable of emulating various machine architectures these two alternatives would incur procurement and/or programming costs in excess of the funding allowance for this study contract.

5.1.2 Computer Aided Design Language

Over the past few years Raytheon has developed a computer aided design system (RAYCAD) for electronic systems (analog and digital), using an in-house, CDC-6700 computer facility. Figure 50 illustrates the scope of the RAYCAD system which, for digital systems, provides: three levels of design verification, two levels of product/hardware design and four levels of test.

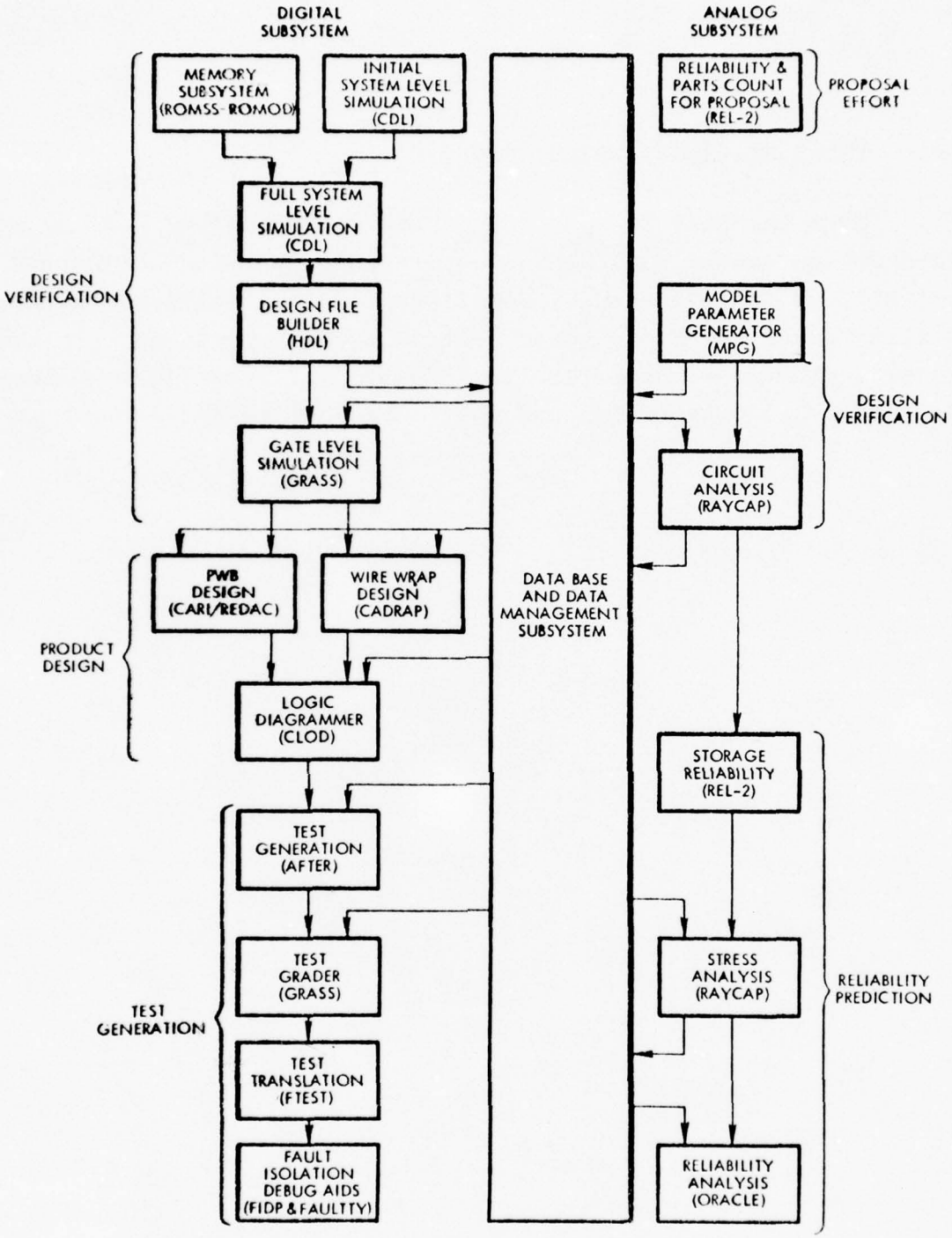


Figure 50 Raytheon RAYCAD System

In the design verification group, the Computer Design Language (CDL) enables digital systems to be simulated in terms of their functional organization, (register, decoders etc.), control algorithms and sequential operation, either in serial or parallel mode and at bit, word or bit-array level. As such, this type of language allows for the specification of a digital system at a level higher than the gate level. It typically may be used to partition a candidate digital architecture into several distinct subunits e.g. register arithmetic and logic unit (RALU), read/write and read-only memories (RAM and ROM), input-output (I/O) units, interrupt controllers etc., so that the designer can study the feasibility and effectiveness of the candidate design.

The benefits of using CDL include:

- Ability to verify architectural integrity of a design.
- Use of CDL as a "design algebra" to easily express ideas and concepts.
- To serve as a common specification language between hardware and software designers.
- To provide documentation deliverable to either customer or other internal working groups.
- Simulation of microcode and use of CDL as a verifier of ROM/PROM code before burn-in.
- Design simulation to provide a performance baseline to serve as a source of data to debug actual hardware.

In view of the above features and their compatibility with the requirements of the macromodule design validation task, CDL was selected as the module simulation tool.

5.2 Computer Design Language (CDL)

Computer Design Language (CDL) is one of a class of languages known as Register Transfer Languages. Figure 51 places CDL in the context of the CAD (RAYCAD) and the user.

The CDL system is divided into three distinct subsystems (TRANSLATE, SIMULATE, and REPORT), and control is passed to the respective subsystem through command directives. Each subsystem has a unique function; the TRANSLATE portion converts CDL source statements into a form (postfix polish string) that the simulator can interpret. The SIMULATION portion executes the polish string in a manner specified through simulation directives and produces a data file to be used by the report generation. The function of the REPORT subsystem is to generate the various types of output required.

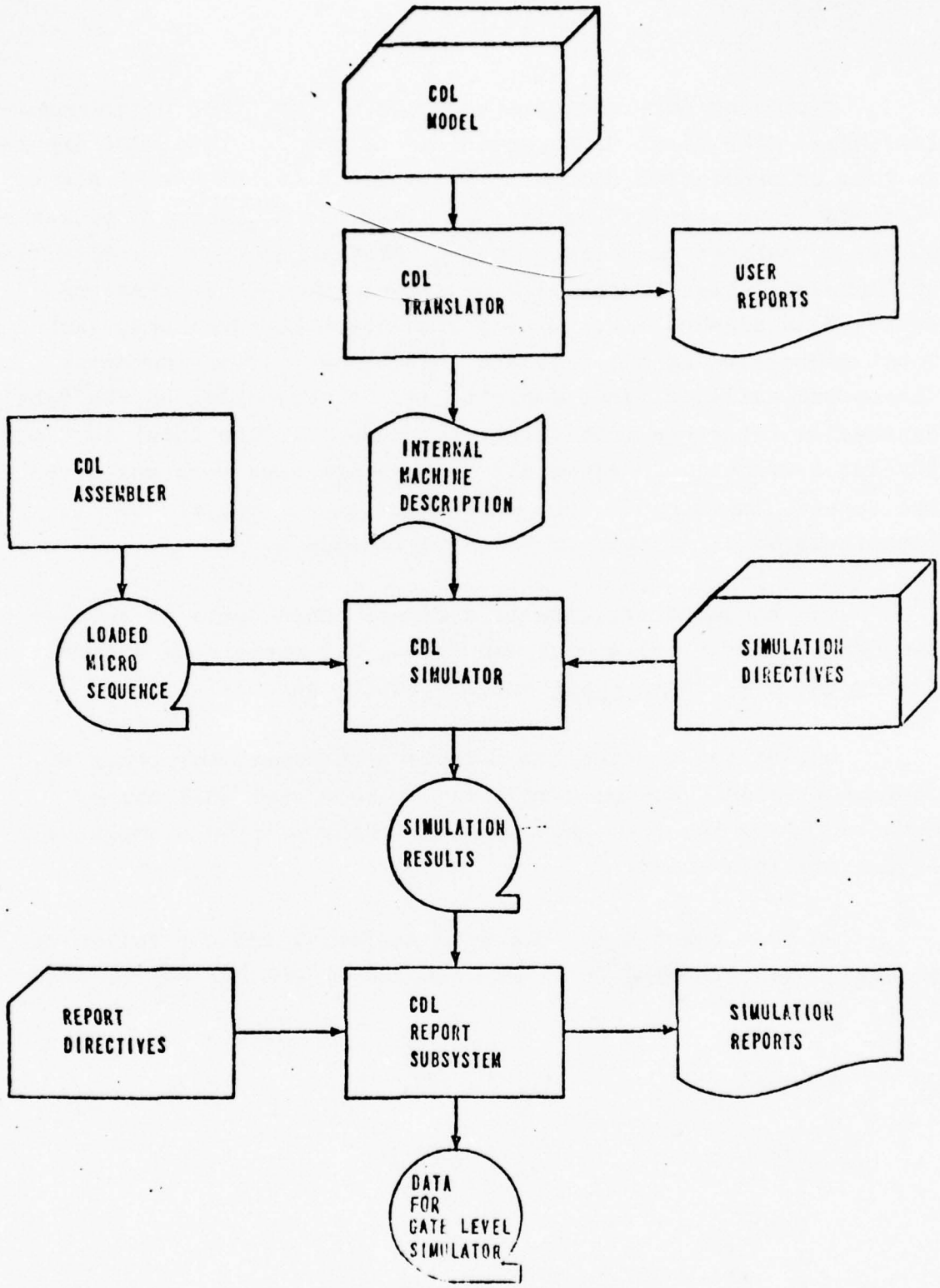


Figure 51 CDL System Relationships

Two major constructs set CDL apart from other programming languages. The first difference lies in the fact that CDL assumes no form of sequential statement execution. In FORTRAN or ALGOL, statement execution is sequential unless the sequence is broken by GO TOs or subprogram calls. In CDL, program sequence is specified by "labels". These labels are unassigned Boolean expressions, enclosed in slashes (/). During simulation CDL evaluates each label expression at the beginning of each unit time increment (hereafter called a label cycle). If the evaluation of the label expression returns a true result (a binary 1), the label is flagged for later execution. After all true labels have been collected for the current label cycle, CDL then executes all operations associated with all true labels SIMULTANEOUSLY.

The second difference is a direct consequence of the first. Because of this simultaneous execution, all operations accomplished during the same label cycle are implicitly parallel.

During the current label cycle a statement may occur which changes a value. The program does not know that this change occurred until the next label cycle. The following example will illustrate this point:

Suppose A and B are declared variables and the following sequences are executed in 1) FORTRAN and 2) CDL during the same label cycle.

FORTRAN	CDL
A=B	A=B
B=A	B=A

In FORTRAN, the result would be: both A and B contain the previous contents of B.	In CDL, the result would be: A contains the previous contents of B. B contains the previous contents of A.

Due to the parallel nature of CDL any attempt to change a value more than once during a single label cycle will result in a random choice of the changes attempted.

5.3 Module Simulations

Due to the effectiveness of CDL in module simulations and the attendant degree of programming required, a group of microcomputer modules was selected for simulation in order to validate the following most significant aspects of the modular design approach:

1. Module performance

2. Timing interrelationships of μCPU-DMAIO & ADAC-DMAIO-RAM

A total of 31,163 lines of code were generated and de-bugged in support of the above module performance and interface timing tests.

5.3.1 Simulation Approach

The major timing mechanism in the CDL language is the clock cycle and the simulation is programmed as advancing in time in clock cycle increments. The basic clock can be further broken down into subintervals called label cycles in order to provide a device for implementing multi-step logic within a basic machine cycle. A

two phase synchronous clock is used in the simulation of each module for compatibility with the staggered addressing/data transfer timing sequence. The sequential aspect of the module/computer being simulated can be observed at each label cycle as though one were looking at a snapshot of all the devices simulated (memories, registers, control lines, flip-flops, etc.) at an instant of time, and it is through observing sequential snapshots that the data can be traced in its flow within the simulation.

The actual timing of data flow within/between modules is measured by the number of clock cycles necessary to complete an operation. In the case of μCPU-1 a 2 MHz clock is used so that each clock cycle snapshot represents a 500 nsec. advancement in time. Consequently, each label cycle approximately represents a 250 nsec increment of time which is the smallest interval observable. The study of faster μCPUs is achieved by changing the time interpretations of the master clock cycle and adjusting counters accordingly.

The concept of standard microbus interface lines between modules, as described elsewhere in this report, is used in the simulation as a means of driving the simulation at execution time. Each of the modules contains READ commands which read the input file of READ records present in the SIMULATE section. The three microbus lines are called MBRW, microbus read/write line, MBDATA, microbus data line, and MBADDR, microbus address line. In most of the modules, the MBADDR is decoded to split off the correct number of bits for the address being transmitted, as well as other bit patterns that represent artificial devices for simulation execution. For example, in the Memory Module, which is written

to simulate both RAM and ROM memories, the MBADDR is decoded as follows:

bits 1-4	DEV = 0 for RAM, = 1 for ROM
bits 5-8	Valid memory indicator
bits 9-16	Actual memory address

The 16 bit MBDATA line is used for data transmission, and the 1 bit MBRW line is set to one for a read operation and to zero for a write operation.

The simulation deck section of the CDL system provides a series of directive statements that define the conditions necessary to test the machine created by the CDL translation phase (model definition phase). It is within this section of the program that input parameters are set for a run. If the module simulated has a control memory, the micro instruction bit patterns are loaded into this memory via *LOAD directives. This provides the means of verifying micro-code before burn-in.

Any memory modules or registers may be loaded in the same manner. Timing counters (constants) are defined as registers in the simulation and are present at this point to represent the alternate timing speeds under study (e.g. A/D conversion delay time). The counter must represent an integer number of master clock cycles.

The user may set up the optional paths of executing the simulation by specifying that programmed switches and interrupts be activated starting at a particular label cycle by use of the *SWITCH and *INTERR directives in this simulation section. The *OUTPUT directive allows the user to specify the CDL components

described in the declaration section which he may want to print out in the $REPORT generation section after the run is completed. The *SIM directive causes simulation execution to start and specifies that it terminate after label cycles.

The report generation section of the CDL system is entered after execution termination. The directives contained in this section provide flexible formats for the printed output.

5.3.2 μCPU Module Simulation

The μCPU module is the central controlling element in each group of modules forming a microcomputer. For the purposes of verifying μBus interface timing and the interrelationship with other modules in a microcomputer group, μCPU-1 was selected for CDL simulation.

The simulation used for this module is based upon the program developed by Hsiu-Jen Lee for the Intel 8080 (Ref. R-17). Lee used the version of CDL available at Raytheon, Bedford Laboratories and the CDC 6700 computer facility for his research work. The resulting program is currently used as a benchmark for checking improvements/updates to the CDL language. Although the simulation itself is much more detailed than is necessary for this study it has nevertheless provided a mechanism for timing some of the basic instructions identified in Section 6 for digital missile guidance and control applications.

To provide the required drivers and the necessary response required for modules associated with the μCPU, a test program was written, (Table 10), and the corresponding CDL verified. Although this program did not run initially, debugging proved to be a

TABLE 10

μCPU-1 SIMULATION TEST PROGRAM

(8080 Assembly Code)

Label	Mnemonic	Memory Location (Octal)	Memory Content (Octal)	Code	Operand	Comment
	LXI D	0100	021			
	⟨ALPHAL⟩	0101	150	LXI	D, ALPHA	Load the register pair D-E Immediate
	⟨ALPHAH⟩	0102	000			
	LXI H	0103	041	LXI	H, BETA	Load the register pair H-L
	⟨BETAL⟩	0104	170			immediate
	⟨BETAH⟩	0105	000			
	MVI C	0106	016	MVI	C, 8	Load register C with '8'
	8	0107	010			
	XRA A	0110	257	XRA	A	Clear Garry
LOOP:	LDAX D	0111	032	LDAX	D	Load register A with M(D-E)
	ADC M	0112	216	ADC	M	Add M(H-L) to A
	DAA	0113	047	DAA		Decimal adjust the result
	STAX D	0114	022	STAX	D	Store A to M(D-E)
	INX H	0115	043	INX	H	Increment the register pair H-L
	INX D	0116	023	INX	D	Increment the register paid D-E
	DCR C	0117	015	DCR	C	Decrement the register C
	JNZ	0120	302	JNZ	LOOP	If not zero, go to loop
	⟨LOOPL⟩	0121	111			
	⟨LOOPR⟩	0122	000			
	HLT	0123	166	HLT		If zero, then stop
ALPHA:	DB	0150	021			
	DB	0151	021			
	DB	0152	042			
	DB	0153	042			
	DB	0154	063			
	DB	0155	063			
	DB	0156	104			
	DB	0157	104			
BETA:	DB	0170	021			
	DB	0171	021			
	DB	0172	042			
	DB	0173	042			
	DB	0174	104			
	DB	0175	204			
	DB	0176	063			
	DB	0177	063			

relatively straightforward task, despite the programmer's lack of experience with CDL.

The test program of Table 10 consists of adding two strings of decimal digits of length eight and replacing the values in the first string with the result of the corresponding sum. The octal microcode and the assembly code are shown in Table 10. By using the *CONTROL directive of the $REPORT section, a compact list of labels that are true for each cycle are listed providing a trace of the control sequences only.

Table 11(A) is an example of a "fetch immediate data to temporary register sequence", and Table 11(B) is an example of tracing the time it takes for the arithmetic logic unit to add the contents of memory location 0170 to register A, leaving the sum in register A. The sequence of operations to complete the store back into memory location 0150 is completed at clock cycle 107, which is an additional 10.5 μsec. For the sample problem described, using this control report in combination with the print out of the various registers, the user can determine that one complete iteration takes 44.5 μsecs and eight times through the loop takes 356 μsecs. All future simulation timings appearing in this report are determined in a similar manner.

TABLE 11

μCPU-1 CDL Coding Examples

(A) Fetch Immediate Data

Clock Cycle	Label Cycle	True Label
36	71	/T1*FIn*R*D(0)/
36	72	/T1*FIn*R*D(1)/
37	73	/T2*FIn*R*D(0)/
37	74	/T2*FIn*R*D(1)/
38	75	/TW*FIn*R*D(0)/
38	76	/TW*FIn*R*D(1)/
39	77	/TW*FIn*R*D(0)/
39	78	/TW*FIn*R*D(1)/
40	79	/T3*FIn*R*D(0)/
40	80	/M3*T3*,'XI*D(1)/

(B) Add Memory to Register A

Clock Cycle	Label Cycle	True Label
81	162	/M1*T4*ALUnPM*D(1
82	163	/OMC*TIM*P(0)/
82	164	/OMC*TIM*P(1)/
83	165	/M2*T1*ALUnPM*D(0
83	166	/M2*T1*ALUnPM*D(1
84	167	/M2*T2*ALUnPM*D(0
84	168	/M2*T2*ALUnPM*D(1
85	169	/M2*TW*ALUnPM*D(0
85	170	/M2*TW*ALUnPM*D(1
86	171	/M2*TW*ALUnPM*D(0
86	172	/M2*TW*ALUnPM*D(1
87	173	/M2*T3*ALUnPM*D(0
87	174	/M2*T3*ALUnPM*D(1

NOTES:

5 Clock Cycles, 10 Label Cycles } = 2.5μsec

7 Clock Cycles, 13 Label Cycles } = 3.25μsec

5.3.3 RAM & (P)ROM Memory Module Simulations

The simulation of the RAM and (P)ROM memory modules as stand-alone simulations was relatively simple in that their function involves the transfer of data between a memory module and μCPU or DMAIO across the μ Bus data lines. The single simulation developed, i.e. MEM1, has the capability of studying both read/write random access memories (RAMs) and read-only memory

(ROMs) modules with slow and fast cycle times (i.e. RAM/ROM-1 and 2 respectively). There are two switches which, under use control, can switch either to a fast or slow memory at a specified label cycle, e.g. -

```
SWITCH    , 17D,  FAST=OFF

SWITCH    , 17D,  SLOW=ON
```

The status of the u Bus lines are used to drive the simulation. MBADDR is decoded into fields:

Bits 1-4, DEV = 0 for RAM, = 1 for (P)ROM
Bits 5-8, valid memory address indicator
Bits 9-16 8 bit memory address

The MBRW line indicates a read or write operation. A write operation is ignored in the case of ROM modules. There is an internal buffer register, IODATA, which is an intermediate register for data during the slow transfer. It is not used in the fast transfers as they are completed in one clock cycle. Timing states are implemented as terminals and their logic is described in the previous section. Reference R-18 contains the complete CDL simulation for RAMs and (P)ROMs.

5.3.4 DMAIO Module Simulation

The specifications for the four modes of operation of the DMAIO are pictured in the functional block diagram and state transition flow chart of the module given in the previous section. The DMAIO contains a ROM control memory called ROM1 that is made up of 16 words of 36 bits. The actual microinstructions for each

possible state of the DMAIO are loaded into memory locations 0 to 11 and the program counter, MIPC1, points to a particular control word address. The micro-instruction pointed to by the program counter is brought into the control memory buffer register, CWD1, at the beginning of every clock cycle. The bit patterns of the micro-instruction are broken out into logic bits (mostly one bit flip-flops) by using the CDL sub-register declarations. The sub-registers critical to the true or false conditions for each machine state of the DMAIO are combined logically using the CDL terminal declarations, which in this simulation, are called STATE1, STATE 2, etc. The true or false conditions of the terminals are evaluated continuously in this simulation since each state appears in a label expression. The micro-code is implemented such that only one state can be true at any point in time. For example, the terminal declaration for machine STATE1 is:

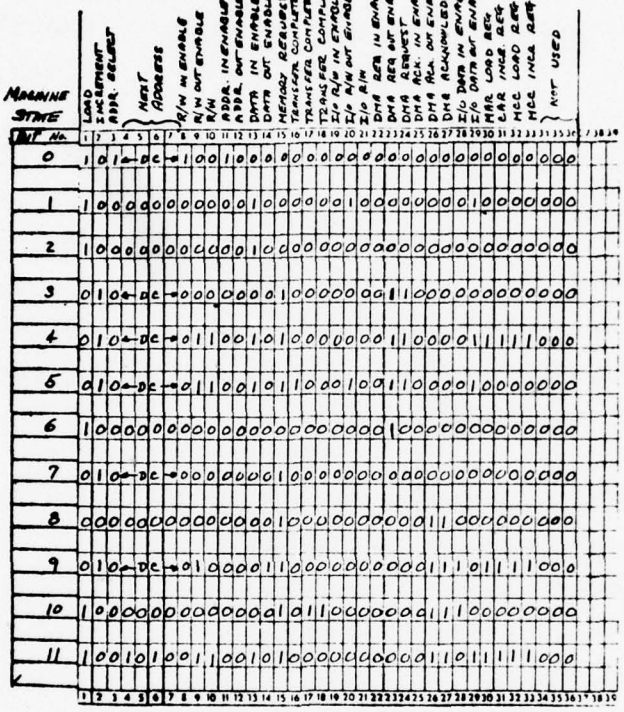
```
TERMINAL, STATE1=LD*DATIN*IORWOT*IODTOT
where * is a logical and operator
        LD is bit 1 of the micro-instruction word
        DATIN is bit 13
        IORWOT is bit 20
        IODTOT is bit 29
```

STATE1 has a true condition only when all four of these subregisters are equal to 1.

The 12 word micro-instruction control program for each machine state is specified in Table 12

TABLE 12

DMAIO MICROINSTRUCTIONS



The use of the machine state terminals and two phase clock condition in the CDL label statements is the mechanism used for effecting the micro-instruction flow during the simulation. (This same approach is used in the implementation of the other modules and will therefore not be repeated in those descriptions.)

As described earlier the simulation events are controlled by reading the status of the μBus lines. In the DMAIO module, the MBADDR is decoded into three fields:

Bits 1-4, DMA, for valid address, DMA-3
Bits 5-7, IODEV, to specify external I/O device with which to interface: ADAC, SDIO, DMAIO2
Bits 8-10, INTREG, codes for internal registers
=1 for CAR, Current Address Register
=2 for TAR, Terminal Address Register
=3 for MCR, Memory Cycle Delay Register
=4 for MR, Mode Register

Varying CPDT from 0 to n allows the user to study the effect of fast and slow conditions for the DMA to access the uBus lines and to study how that condition slows up an I/O transfer. All of these counters are artificial devices to allow this module to be executed on a stand-alone basis with the user estimating delays due to device interfaces.

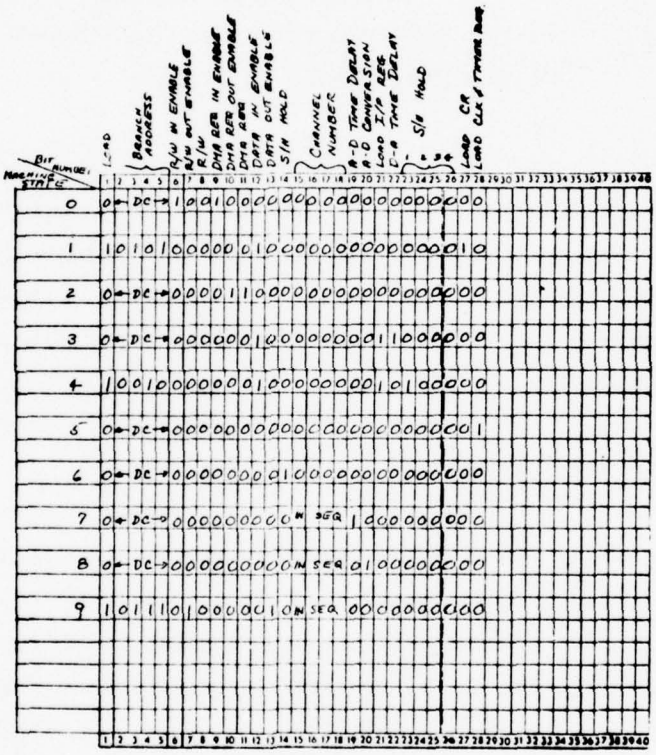
5.3.5 ADAC Module Simulation

The ADAC module simulation was based on the state transition diagram given in Section 4. There are two independent modes of operation: 1) transferring analog information from the analog input channels through the A-D conversion process to the μBus data lines, MBDATA, and 2) transferring digital data from MBDATA through the D-A conversion process to the analog output channels. The steps in each of the processes are implemented using delay counters that can be set by the user at execution time to study the stable state of the data at the interfaces and the critical time elements in the transfers.

Like the DMAIO module described earlier, the ADAC contains a small control microprocessor which monitors the events necessary to move the data through the device. The ROM, called ADROM, contains 15 words of 36 bits that is initially loaded with the control program shown in Table 13 which represents the 10 machine states.

TABLE 13

ADAC MICROINSTRUCTIONS



The instruction execution is determined by the value of the program counter, ADPC. The use of the CDL terminal declarations for determining the true condition of any transition state and

the appearance of these terminal names in label statements effect the microinstruction flow of the simulation. Switches allowing only D-A or A-D transfers to occur and interrupt declarations which allow for automatic scheduling of interrupts are simulation devices used to trigger events in the simulation execution. The external control lines to the interface units are all included, but the acknowledgement of requests is handled artificially in this stand-alone module.

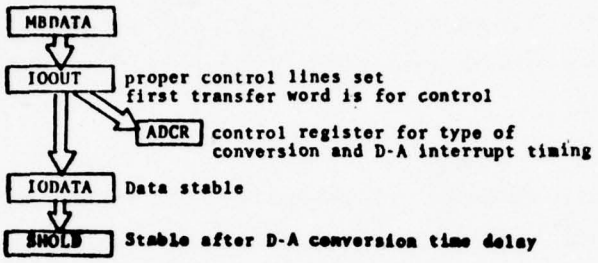
There are several intermediate holding registers in the simulation which can be observed in the output to snapshot the sequence of logic for the data transfers. There are also many special counters which effect the necessary delays for the conversion process. Figure 52 (A) shows the data flow for a D-A interrupt. Data is put on MBDATA by the DMAIO; the ADAC device is requested. The first word contains an indicator for D-A, and bits 13-16 contain the count of clock cycles for the next D-A interrupt. Transition state 2 cause cycling of the DMA requests for data to be put onto MBDATA until the entire block of data from memory has been moved to the analog output channels. Figure 52 (B) shows the data flow for data moving from the analog input channels to the μBus data lines. The interrupt scheduling for the A-D is an input parameter to this module since it is the ADAC itself which institutes the DMA request.

The following timing parameters are built into the ADAC module.

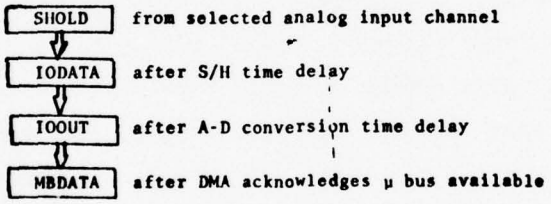
ADTIM - Number of clock cycles between A-D transfers
(setting ADTIM=4000D is equivalent to 2 msec)

DATIM - Number of clock cycles between D-A transfers. Bits 13-16 in MBDATA for I/O initialization.

ADTDC - Number of clock cycles for A-D conversion time delay.



(A) D-A



(B) A-D

Figure 52 ADAC Module Data Flow

DATDC - Number of clock cycles for D-A conversion time delay.

SHTDC - Number of clock cycles for sample and hold amplifier (S/H) time delays.

Varying these parameters allows for the study of the effect of component delays on the module's operation.

5.4 Microcomputer Configuration Simulations

To provide a more realistic simulation for timing of events and designing logic interfaces, two simulations are described which combine the CDL module simulations described in the previous section.

5.4.1 μCPU/DMAIO/μBus/RAM Configuration

The first combined simulation allows the μCPU-1 CDL module described in subsection 5.3.1 to interface with the DMAIO module described in section 5.3.2. The purpose in combining these two modules was to study the timing delays which will occur in the arithmetic calculations of the microprocessor when an I/O module requires access to the μBus lines to transfer data to and from a RAM module.

Figure 53 shows the effective microcomputer configuration simulated.

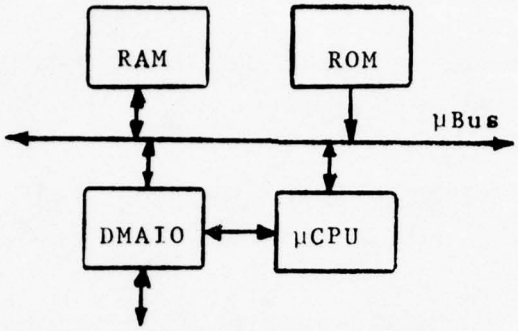


Figure 53
Modular Microcomputer Simulation, Configuration 1.

For DMAIO transfers the μCPU-1 must go into a HOLD state at the next possible machine cycle *after receiving a memory* request from the DMAIO, it must send a request acknowledge, CPMAK, back to the DMAIO when the bus lines are available and must come out of the HOLD state upon receipt of the end of block, EOB, interrupt from the DMAIO, assuming a burst transfer.

DMAIO Initializing - The microprocessor initializes the DMAIO registers prior to such data transfers between memory and the various I/O devices by "writing" into assigned DMAIO registers the beginning and end of memory block addresses where the data will be stored or where it will be accessed, the value of the memory cycle delay, and the mode of the transfer operation.

ADAC Initializing - The microprocessor also uses the DMAIO module to initialize the interfacing I/O module. In the case of the ADAC unit, it does this by transferring a control word on the data bus indicating the number of clock cycles between D-A transfers of data to the appropriate guidance module. The A-D interrupt clocks are also initialized by the μCPU at the beginning of a run. The ADAC stores the MBDATA word in its own control register, ADCR, where it is decoded into interrupt cycle times and I/O mode.

This simulation was tested by executing a combination of the tests run on each module's stand-alone simulation. The μCPU-1 test program (consisting of adding two strings of decimal digits) is executed by the CDL simulation of the μCPU as an iterative loop, taking 387.25 μsecs for 8 passes. At a predefined label cycle the DMAIO interrupt variable ICON is switched on and the initialization and execution of a block transfer is forced to take place by reading the contents of the μBus lines using the *READ commands. After the end-of-block indicator is set, the interrupt to read the lines is disabled and the μCPU arithmetic calculation drops out of the HOLD state and continues.

In an effort to improve the running time of this simulation, many of the label statements and terminal statements that are not necessary for the test case being executed were deleted from this version of the I8080/DMAIO implementation. This was an economic decision due to the high cost of running the over-specified (for our purposes) microprocessor simulation.

Test Results - The results obtained from the μCPU-1/DMAIO interface simulations are summarized in Table 14 below. Both modules are driven by the 2MHz μCPU-1 clock. (A 3:1 speed improvement would be obtained with the 6 MHz μCPU-2 clock).

TABLE 14
μCPU-1/DMAIO INTERFACE TIMING
(2 MHz Clock)

INTERFACE	TIME-DELAY (μsec)	COMMENTS
DMAIO to μCPU μBus Access	2.0 AV.	Per access average for 10 accesses.
DMAIO to DMAIO, RAM to RAM Word Transfer	4.9/6.4	Per word average for 20-word block. 200 nsec/1 μsec RAM cycle times.

The DMAIO to μCPU access time was determined on the basis of the average number of label cycles between the time that a memory request was made and the time that the DMAIO received an acknowledgement. A sample of 10 requests indicated, on average, a delay of 8 label cycles or 2.0 μsec.

The DMAIO to DMAIO, inter-microcomputer path for RAM to RAM data transfers was exercised every 100 μsecs, transferring a block of 20 words each time.

This test using the combination simulation is a more realistic approach than inputting the central processor delay time, CPDT, as was done in the DMAIO stand-alone module. Observing the effects of DMAIO memory requests on the computation speed of the μCPU makes the results more responsive to the number of words transferred, the speed of the memory and the frequency of the memory requests.

5.4.2 ADAC/DMAIO/μCPU/μBus/RAM Configuration

To verify the timing and interface of modules interconnected to form a whole microcomputer the ADAC and DMAIO modules were coupled together and interfaced with the μCPU, μBus and RAM modules. The μCPU was then exercised with the test program as an iterative loop to simulate normal activity of the μCPU while A-D/DMA and DMA/D-A input-output transfers were taking place. Figure 54 show the effective microcomputer configuration simulated in this manner.

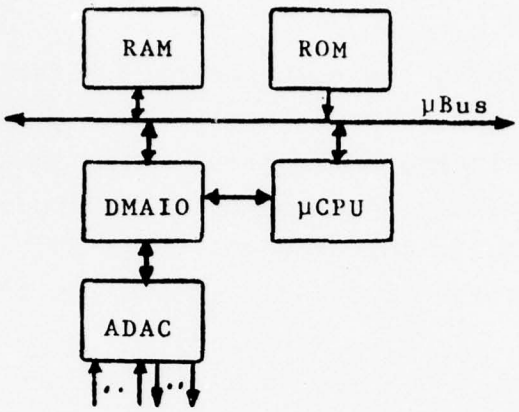


Figure 54
Modular Microcomputer Simulation, Configuration 2.

The DMAIO is the key module for all input/output activity in the various microcomputer configurations. By combining the DMAIO module with the logically complex ADAC interfacing I/O module and using the associated μCPU and RAM interface timing, we are able to realistically assess the timing criteria for data transfer to and from main memory and an I/O device.

The task of combining these modules uncovered logical control errors in the original modules that would not have been detected by running them in their stand-alone state. Most of these errors were the result of incorrect implementation of timing the request and acknowledgement flags. All the timing parameters for the ADAC module are included in this simulation, and all but one of the timing parameters for the DMAIO module are included also. The parameter removed is the artificial counter for the number of clock cycles for a transfer of data to or from an I/O device to be completed, IODT. The IOTRCP, I/O transfer complete, flip-flop is

correctly set by the simulation logic of the DMAIO in this model. A complete listing of this CDL simulation is included in Supplement 1 for those interested in the implementation logic.

A-D & D-A I/O - To test all paths of the simulation, a run was put together which transferred four words of data from memory locations RAM1 (145-150)) to the analog output channels using the D-A logic. After this the run is set up to switch on the first A-D interrupt at label cycle 71. This will re-initialize the internal registers of the DMAIO so that four analog input channels' contents will be stored into memory locations RAM1(111-114). The contents of the four channels are the respective channel numbers 1-4. Reference R-18 contains output reports demonstrating the movement of the data through both of these transfers. Most of the timing delay parameters were set to unrealistically small values to generate this output for economic reasons since we were testing data flow logic, not timing parameters. Results of the timing studies are included in the following section.

The execution of the simulation starts by switching POWER (on) which sets all internal registers, counters, buffer registers, and the control program counter to zero. The START switch causes micro-instruction zero to be loaded into CWD1, which causes state zero to be true. State zero reads the μBus lines and the simulation follows a path controlled by the contents of MBADDR thereafter.

The following timing parameters are built into the DMAIO module.

CPDT - Number of clock cycles between the time the DMA makes a memory request of the μCPU and the acknowledgement of that request (i.e. how long before the μbus lines are available).

DMADT - Number of clock cycles between the time the I/O device requests memory of the DMA and the time that request is acknowledged.

DM^2DT - Number of clock cycles between the time that DMAIO #1 requests memory of DMAIO #2 for a RAM to RAM transfer and the time that request is acknowledged.

IODT - Number of clock cycles for a transfer of data to or from the I/O device to be completed. This counter is used in lieu of the actual device being simulated. It is unused in this combined simulation.

Test results - Table 15 shows the ADAC/DMAIO interface timing results based upon a 1 μsec memory cycle and 2 MHz (500 nsec) clock.

TABLE 15

ADAC/DMAIO INTERFACE TIMING

(2 MHz Clock)

INTERFACE	TIME DELAY (μsec)	COMMENTS
RAM-DMAIO-D/A (Per Word/Channel)	10.4 av.	Average for 4 wds/chs. 5 μsec. D-A conv. time.
A/D-DMAIO-RAM (Per channel/word)	8.1/11.7	Average for 4 chs/wds. 2.5/6.0 μsec A-D conv. time.

6. MICROPROCESSOR ARCHITECTURES VS PERFORMANCE

As part of the missile design application task, to determine the best fit of microprocessor architecture and instruction set to each of the four major missile functions, namely: steering command generation, seeker head control, autopilot and warhead fuzing, the following candidate processor architectures and associated support software were exercised using the algorithms and program modules developed in the Phase I and II studies.

TABLE 16

TARGET MICROPROCESSOR ARCHITECTURES

	PROCESSORS						
	μCPU-1&-2					μCPU-3&-4	
FUNCTION	8080	Z-80	M6800	6100	3000 8080	PDP-11/34	AN/UYK-20
Steering Command Generation	-	-	-	-	-	X	X
Seeker Head Control	X	X	X	X	X	-	-
Autopilot	X	X	-	X	X	-	-
Fuzing	X	X	-	X	X	-	-

In the low-cost 8 - bit "CPU-on-a-chip" category of CPU-1 the Intel 8080A (Ref. R-12), and Zilog Z-80 (Ref. R-19), processors were chosen as two practical examples of general register architectures, while the Motorola M6800 (Ref. R-20), served as a single accumulator counter part. The Intersil/Harris 6100 12-bit "CPU-on-a-chip" (Ref. R-21), was exercised to determine the effect of increasing the word length for the same missile functions.

For the μCPU-2, the high-speed, Signetics bipolar equivalent of the 8080 was used, since this employed a bit-slice, register arthimetic and logic (RALU) architecture driven by a micoprogrammed control unit, in accordance with the modularity concept of the missile microcomputer family.

Target architectures selected for μCPU-3 and-4 were the Digital Equipment Corp. PDP-11/34 (Ref. R-22), and Navy/Univac AN/UYK-20 (V), (Ref. R-13 and R-14), since these were both mature, well-supported and widely used machines with internal register files of: 8, for the PDP-11/34, and 16 or 32 words for the AN/UYK-20.

The availability of FORTRAN IV compilers, together with CMS-2 and SPL/1 for the AN/UYK-20 (V), in addition to symbolic assemblers, provided the opportunity to undertake more extensive coding of the Phase I and II algorithms within the Phase III budget. This in turn enabled the effectiveness of the higher order languages (HOLs) to be determined for missile guidance and control applications.

The use of direct memory access (DMA) for all input - output (I/O) transfers in the missile microcomputer family provided independence from the peculiarities of the various target processor I/O mechanisms and associated instructions.

Details of the above computer architecture and programming exercises are discussed in Sections 6.1 and 6.2. The approach in each case was as follows:

1. Code the applicable Phase I and Phase II program modules using the target machine instruction set.

2. Determine the appropriate computational delays in terms of number of machine/clock cycles and actual run times for the corresponding μCPU-memory device technology employed.

3. Determine the program and data base sizes.

4. List the instruction types used.

5. State the accuracy of the computed results.

6. Note improvements, if any, in processor architecture and instruction set which would simplify the software task and improve performance.

Programs coded were not executed on the respective target computers since the objective was to take a relatively quick look at many options and determine a fit of function to architecture worthy of more time and expense in achieving actual debugged/working programs.

All coding/estimating was performed by programmers experienced in both the missile function disciplines and the coding of the processors identified.

Program listings for $_{\mu}$CPU-1&2 and $_{\mu}$CPU-3&4 are contained in Reference R-23.

6.1 $_{\mu}$CPU-1 & $_{\mu}$CPU-2 Architectures

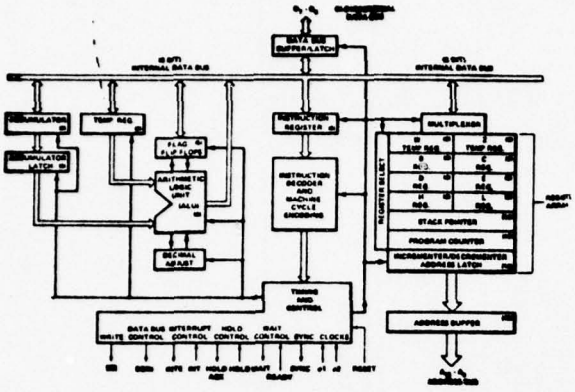
Significant features of the four microprocessors evaluated for $_{\mu}$CPU-1 and $_{\mu}$CPU-2 are given in Table 17. Functional block diagrams are shown in Figure 55.

TABLE 17

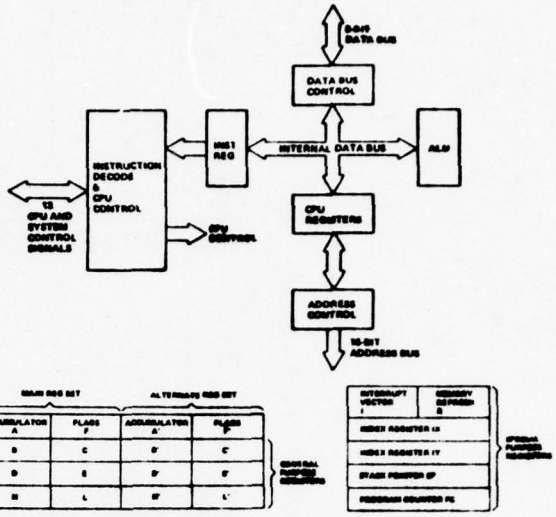
μCPU-1&2 TARGET MICROPROCESSOR ARCHITECTURAL FEATURES

MACRO MODULE	TARGET PROCESSOR	WORD SIZE (BITS) INSTRN.	DATA	REGISTERS GENERAL	DEDICATED	ADDRESSING MODES	INSTRN. SET	CYCLE TIME (nsec)
	8080A (AM9080A-2DM)	8/16/24	8	3 x (8+8)	ACC (8) PC (16) SP (16) SR (8)	Immed. 8/16 Direct 8/16 Indirect	78 16+16 R-R add. No mpy or div.	380
μCPU-1	Z-80	8/16/24	8	6 x (8+8)	2xACC (8) PC (16) SP (16) 2xSR (8) 2xIR (16)	Immed. 8/16 Direct 8/16 Indirect Indexed Relative (PC)	158 16+16 R-R add No mpy. or div.	400
	M6800	8/16/24	8	NONE	2xACC (8) PC (16) SP (16) IR (16) SR (8)	Immed. 8/16 Direct 8/16 Indirect Relative (PC)	72 No mpy or div.	1000
	HM-6100 (PDP-8/E)	12	12	NONE	ACC (12) TR (12) PC (12)	Direct 12 Indirect Indexed	73 No mpy or div.	325
μCPU-2	3000/8080	8/16/24	8	3 x (8+8)	ACC (8) PC (16) SP (16) SR (8)	Immediate Direct M&R-R Indirect	80 16+16 R-R add Incl. mpy & div.	150

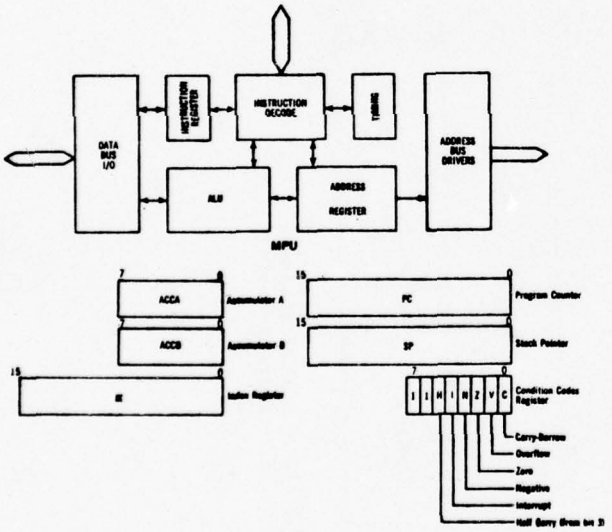
Legend
ACC - Accumulator SR - Status register
PC - Program counter IR - Index register
SP - Stack pointer



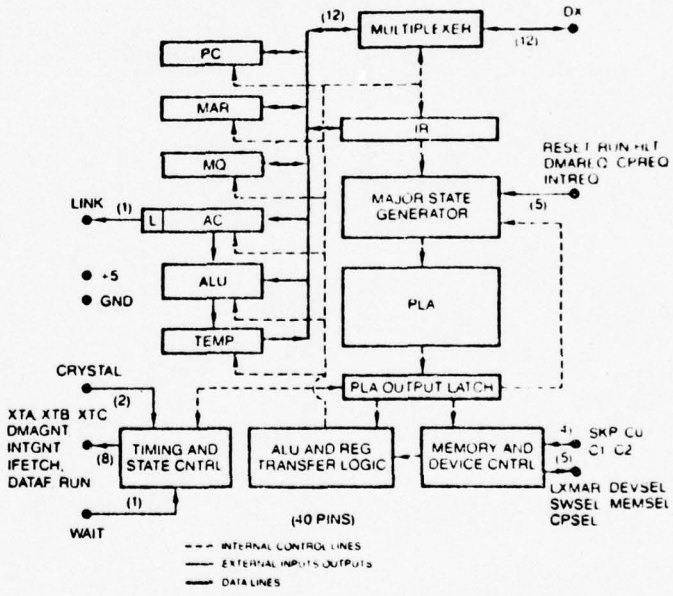
(A) Intel 8080A



(B) Zilog Z-80



(C) Motorola M6800



(D) Intersil/Harris 6100

Figure 55 Target Microprocessor Architectures,
µCPU-1 & µCPU-2, Block Diagrams

6.1.1 Head Control

As described in the Phase I Report (Ref. R-1), and Sect. 3, control of a target seeker gimballed platform involves two distinct operating modes: (1) Head Aim - the initial slaving of the head to position commands ($V\theta c$, $V\psi c$) from the AWCS via the umbilical and microcomputer serial digital input-output interface (SDIO module), and (2) Tracking and Stabilization-post launch control of seeker pointing angles (θs, ψs), stabilized against missile body motion, through boresight error commands, (εp,εy), derived from the target seeker and steering command generation microcomputer (SCGC), and inertial rates from the pitch and yaw gimbal rate gyros, (ωsp ωsy).

Microcomputer Configuration

Figure 56 shows the compatible microcomputer configuration for 2-axis (pitch and yaw) seeker head control applicable to Class I and Class II missiles (see system diagrams: Figures 10 and 11). The μCPU-1/-2 candidate microprocessor is supported by: read/write and read-only memory modules, RAM/(P)ROM, (compatible with the processor under evaluation); one DMA/ADAC combination for gimbal sensor and torquer analog interfaces; one DMA/SDIO pair for AWCS (umbilical) or SCGC serial digital interface, together with an optional hardware multiply module (HMPY-1) for throughput enhancement, subject to the results of the following microprocessor performance evaluations.

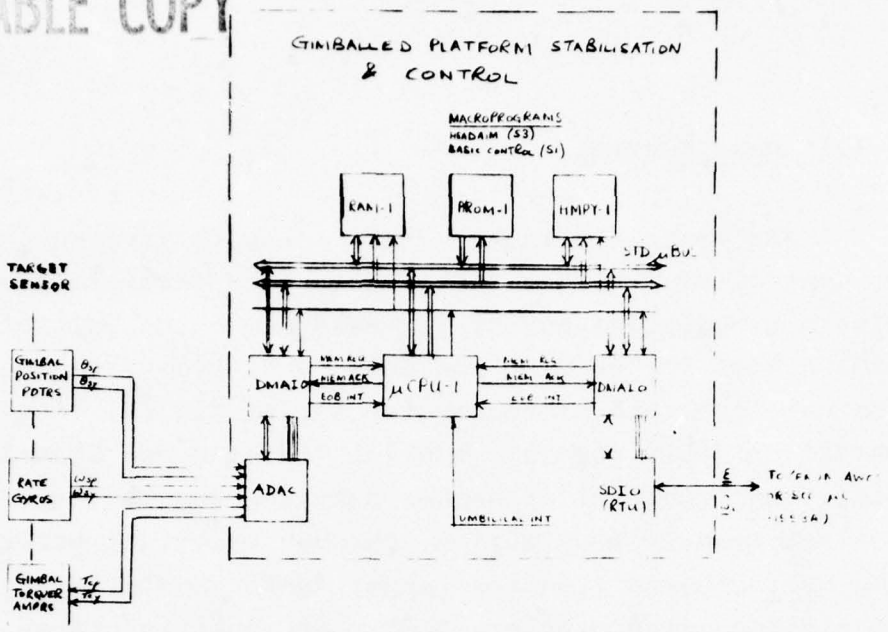


Figure 56 Head Control Microcomputer Configuration, Block Diagram.

Microcomputer Operation

The functional operation of the head control subsystem is determined by the operating program in (P)ROM as illustrated in the flow chart of Figure 57.

Initialization - Application of DC power to the microcomputer modules is accompanied by an automatic reset command which clears the program counter in the microprocessor and, in turn initiates the fetching of the first instruction from (P)ROM, which is the first of a sequence of initializing orders to clear all working registers in the μCPU.

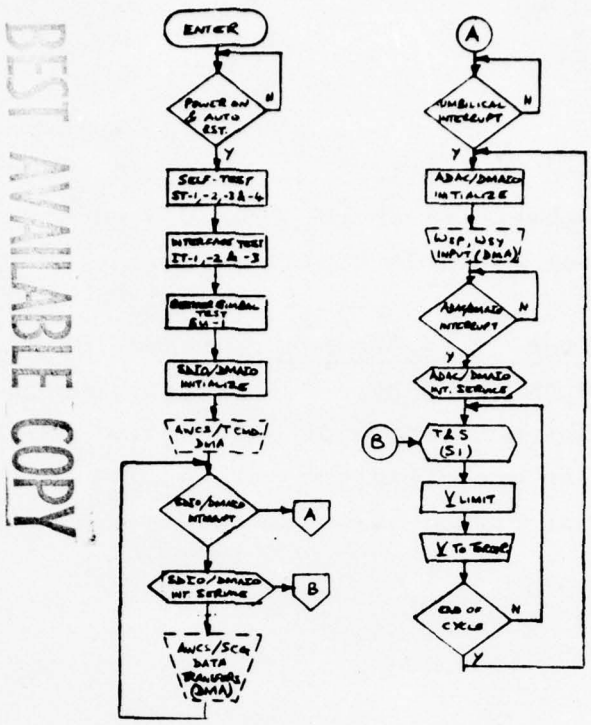


Figure 57 Head Control Subsystem Operational Flow Chart

Self-Test - At the completion of the μCPU initializing program, test modules are called to verify the proper function of μCPU, memory, HMPY, DMA/ADAC modules and the gimballed platform, based on programs identified in the Phase II Report. These test results are stored in an I/O buffer area of the RAM module for subsequent output to the AWCS upon command.

AWCS-Umbilical Interface - When the latter tests have been completed, the DMAIO modules are initialized in readiness for AWCS-umbilical command, data and status word transfers to/from RAM through the SDIO, and to input gimballed platform data to RAM via the ADAC module. ADAC data is stored in a fixed set of RAM locations. Similarly, SDIO data is input and output to/from I/O buffer locations in RAM accessible through the subaddress and word count fields of the AWCS 1553A command message. Test results are

output upon command by the AWCS computer and bus controller, verifying proper operation of the DMA/SDIO module in the process through transmittal of 1553A status words.

Head Aim/Lock-On Before Launch - Receipt of seeker head aim commands from the AWCS and head pointing angles from the platform is signalled by the occurrence of SDIO/DMAIO and ADAC/DMAIO, interrupts, the latter initiating the calling of the head aim program module (S3) Figure 58 described in the Phase I report.

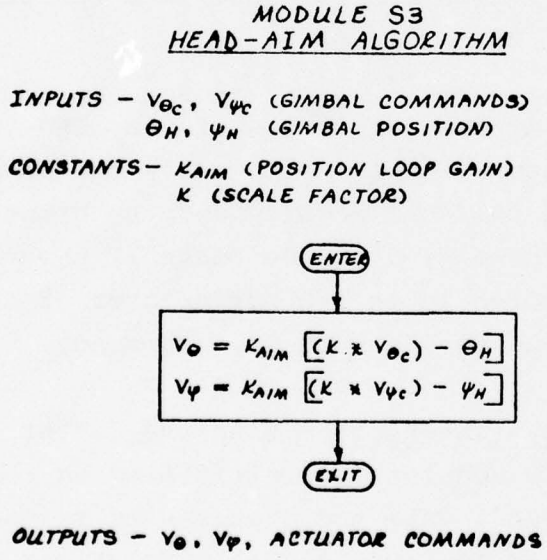


Figure 58 Head Aim Algorithm.

The head aim program is repeated every 2 msec for both missile classes, using cyclic inputs of θs and ψs from the ADAC sampled synchronously with the gyro reference frequency, and updates of Vθc and Vψc from the AWCS. Computed gimbal torquer commands, Vθ and Vψ are output from RAM via the DMA/ADAC modules, (i.e. the D-A converter section), by program initiation. Feedback of seeker pointing angles to the AWCS, to determine target lock-on, occurs when requested by the AWCS via a 1553A transmit command to the SDIO. (See Figure 59 for SDIO/DMAIO interrupt service routine flow chart). The head aim sequence continues until the umbilical is disconnected whereupon an interrupt to the μCPU initiates the calling of the track and stabilization program (S1) Figure 60.

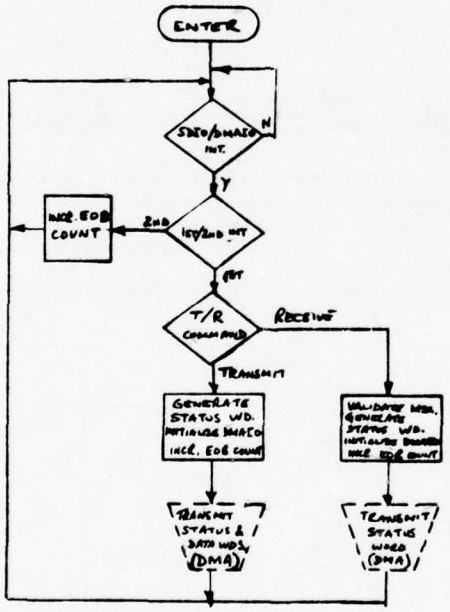


Figure 59 SDIO/DMAIO Interrupt Service Routine, Flow Chart

Track and Stabilization - ADAC inputs continue uninterrupted, and boresight error ($\underline{\varepsilon}$) commands are received by the DMA/SDIO from the steering command generation microcomputer (SCGC) in place of the former head aim commands from the AWCS. Based upon data derived in the Phase II study, the maximum allowable computational delay for Class I and II missile head stabilization is 800 and 600 μsec per channel respectively, i.e. from the time of sampling the gimbal rate sensor to the output of the compensating torquer command. The microprocessors evaluated were therefore judged on their ability to execute the tracking and stabilzation algorithm in the time available after A-D and D-A conversion and transfer to/from RAM. The self-test and head aim functions are relatively non time-critical, e.g 1 sec and 8 to 16 msec respectively.

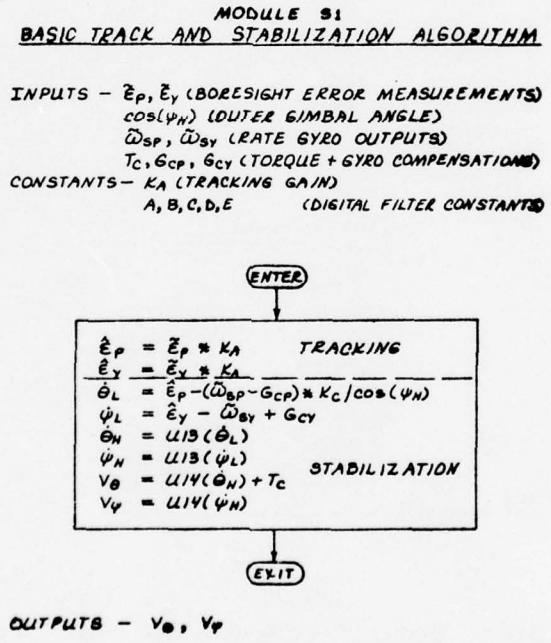


Figure 60 Track and Stabilization Algorithm

Microprocessor Performance

Table 18 lists the results of coding the head aim and track and stabilization algorithms in the assembly language of each of the five microprocessors evaluated.

μCPU-1 - From the data obtained it can be seen that while all μCPU-1 configurations have adequate throughput for the head-aim position servo function, the hardware multiply module (HMPY-1) is essential to achieve the required throughput and computational delay with the N-MOS 8080, Z80 and 6100 processors for the tracking and stabilization function. The 6800 with its slow clock cycle time (1 msec) and lack of double precision operations is marginally suitable for a Class II missile, with the aid of a hardware multiply module.

μCPU-2 - Using performance data derived from the Signetics 3000-based 8080 emulator, computational delays for tracking and stabilization were decreased by a factor of 5 approximately, compared to the N-MOS 8080. However, this speed improvement is not required for Class I and II missile head control since the lower-cost N-MOS 8080 with a hardware multiplier module meets the performance requirements. For Class III missiles employing gyro error and geometry compensation algorithms, the availability of a five-fold throughput improvement provides the necessary modular growth capability through the substitution of μCPU-2 for μCPU-1 and the replacement of the RAM/ROM-1 memory modules with high speed RAM/ROM-2 equivalents incorporating the additional compensation algorithms.

TABLE 18

HEAD CONTROL VS MICROPROCESSOR PERFORMANCE

FUNCTION	μCPU-1 (N-MOS) 8080A +SMPY	8080A +HMPY	Z80 +SMPY	Z80 +HMPY	6800 +HMPY	6100 +HMPY	μCPU-2 (BIPOLAR) 8080 +SMPY	8080 +FMPY	8080 +HMPY
MICROPROCESSOR CLOCK (nsec)	380	380	400	400	1000	325	150	150	150
HEAD AIM (PER CH.)									
No. of Clock Cycles	904	290	893	289	142	308	-	-	-
Computational Delay (μsec)	344	110	357	116	142	100	-	-	-
TRACKING & STABILIZATION (PER CH.)									
No. of Clock Cycles	3453	871	3435	890	476	828	1967	893	647
Computational Delay (μsec)	1312	331	1374	356	476	269	295	134	97
TOTAL PROGRAM SIZE (BYTES)	714	800	700	778	1259	361 WDS.	714		800

LEGEND:
- SMPY Software multiply subroutine
- FMPY Firmware multiply microprogram
- HMPY Hardware multiply macromodule.

Multiply Operations - In the course of the μCPU-1 and -2 microprocessor performance evaluations, six forms of multiply mechanization were reviewed and their respective execution times determined. Table 19 summarizes the results.

TABLE 19

μCPU-1&-2 MULTIPLY METHODS Vs SPEED

MICROCOMPUTER CONFIGURATION	TYPE OF MULTIPLY	OPERANDS (BITS)	EXECUTION TIME (μSEC) MEM-MEM	REG-REG
N-MOS 8080	Software	8 X 16	143	126
N-MOS 8080	μBus HMPY-1	8 X 16	34.2	17.1
Bipolar 8080	Software	8 X 16	29.0	24.3
Bipolar 8080	Firmware	8 X 8	31.4	36
Bipolar 8080	μBus HMPY-1	8 x 16	9.3	4.65

From the above results it is apparent that the N-MOS 8080 with HMPY-1 module is more effective than the bipolar 8080 emulator with a multiply instruction implemented in conventional firmware (i.e. microcode). Thus the CPU-on-a-chip 8080 can be made quite potent in multiply rich applications where the 5:1 speed improvement in other operations is not required.

Instructions Used - Of the 78 and upwards instructions available in the micro processors evaluated for the head control functions, no more than 18 different types were required. In the case of the 12-bit 6100/PDP-8E processor, 7 different types sufficed. Table 20 lists the instructions exercised for each processor type. A total of 4 8 X 16-bit multiplies are required for head aim and 16 8X16-bit multiplies for tracking and stabilization.

Computing Precision - ADAC module A-D and D-A conversions were assumed to be 12 bits 2's complement. Double precision (16-bit) addition and subtraction was performed in the coding process, with 16X 8-bit multiplications yielding 24-bit products, the least significant 8 bits of which were discarded.

Microprocessor Improvements

During the course of coding the head control functions on the candidate microprocessors the following improvements were noted as aids to programming and throughput.

1. 6800 - Provision of 16-bit operations and direct memory to memory move instructions, since most accumulator loading was intermediate to storage in another memory location.

TABLE 20

HEAD CONTROL-INSTRUCTIONS EXCERCISED

INSTRUCTION CATEGORY	MICROPROCESSOR 8080A	Z80	6800	6100
Data Transfer	LHLD	LD	LDAA	DCA
	SHLD	EX	LDAB	
	MVI	PUSH	STAA	
	MOV	POP	STAB	
	LXI	BIT		
	XCHG			
	PUSH			
	POP			
Arithe-	DAD	ADD	ADDA	CLA
metic	ADC	ADC	ADCA	TAD
& Logical	SUB	SUB	SUBB	CIA
	SBB	SBB	SBCA	AND
	ANI	SBC		
	RAL	INC		
		RLA		
Program	JP	JR	BMI	SZA
Control	JNC	CALL	BGE	SNA
	CALL	RET	BSR	
	RP		RTS	
			RTI	
			WAI	

2. 8080 and Z80 - Provision of a double subtract instruction which does not require resetting the Carry Flag prior to execution. Also double negate and direct Z-80 memory to memory block instructions. Z-80 memory to memory block moves require that several of the registers be preset which is undesirable for memory blocks less than 4 bytes.

3. 6100 - Provision of direct memory to memory move instruction.

Subsystem Performance

For system performance evaluation purposes, the head control loop time delays are summarized in Table 21 using the N-MOS 8080A with HMPY-1 module. The total delay times are well within the 800 and 600 μsec allowances for Class I and Class II missiles respectively.

TABLE 21

HEAD CONTROL-TRACKING & STABILIZATION LOOP DELAYS

ELEMENT	DELAY (μSEC) CLASS II	DELAY (μSEC) CLASS II	COMMENTS
DMA/ADAC Input	46.8	46.8	All sensors
T & S Program	331.0	331.0	One channel
DMA/ADAC Output	5.2	5.2	All torquers
t_{STAB} (μSEC)	383.0	383.0	

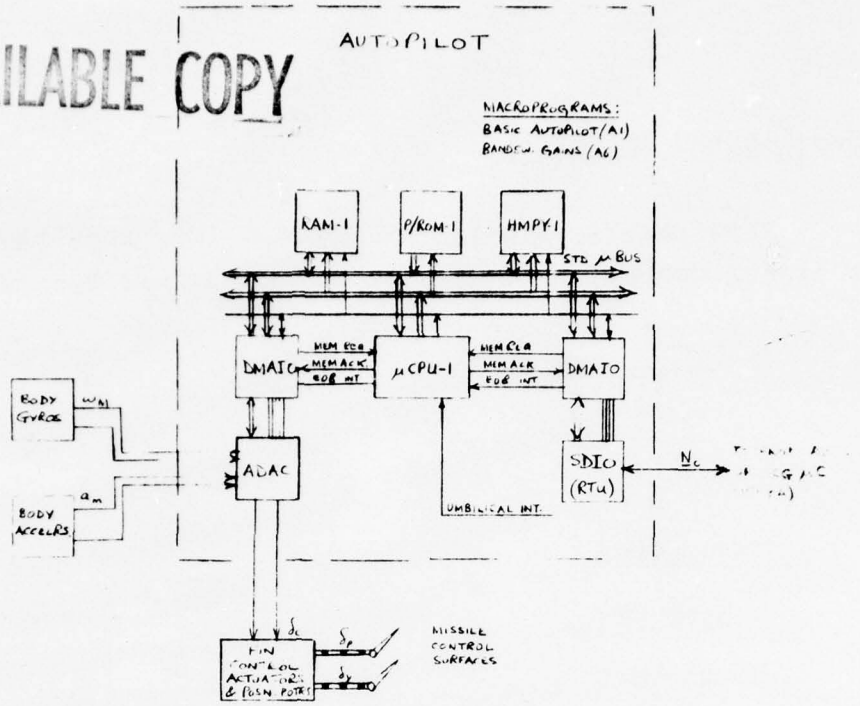
6.1.2 Autopilot

Following the application of the five 8 and 12-bit target microprocessor architectures to gimballed platform head control, the N-MOS and bipolar 8080, Z-80 and 6100 processors were selected for evaluation in the role of an autopilot computer for Class I and II missiles. The 6800 was not considered due its low throughput.

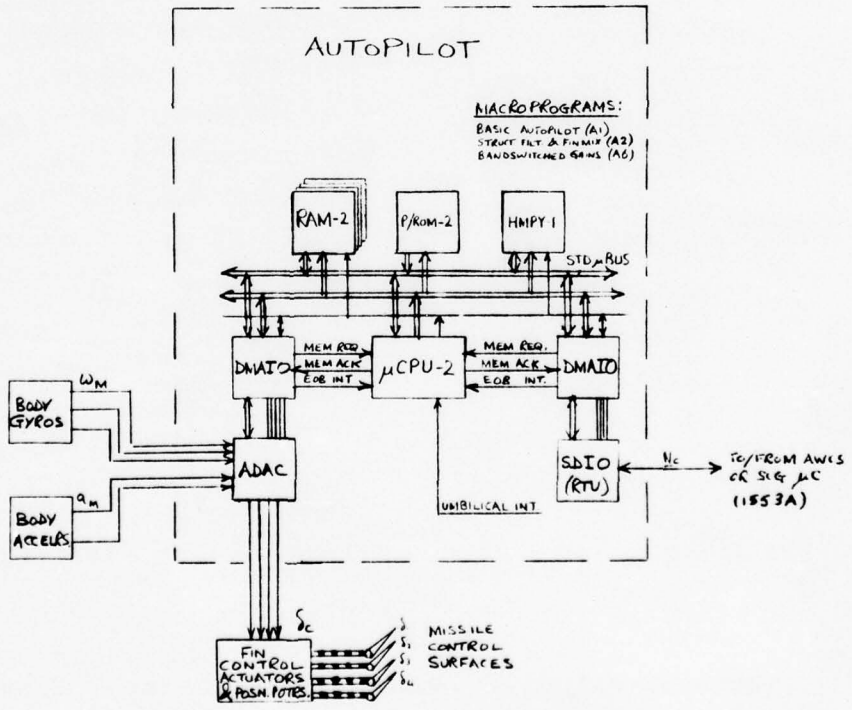
Microcomputer Configurations

Figures 61(A) and (B) show block diagrams of the modular microcomputer configurations for each of the two missile classes respectively. These computers are similar in many aspects to the previous head control computer, (Fig. 56) since missile body mounted rate gyros replace the gimballed platform equivalents together with additional accelerometer inputs, and fin control actuators become the counterparts of the gimbal torquers.

The Class I flight control system employs a simple two-axis (pitch and yaw) band-switched autopilot, whereas the Class II system incorporates three-axis control of four orthogonal fins with 4-pole structural filters to compensate for missile body bending effects. The basic autopilot algorithm (A1) is common to both classes but the Class II missile constitutes a growth version through the addition of a third control channel, (roll), structural filters and fin mixing algorithms (A2).



(A) CLASS I MISSILE AUTOPILOT

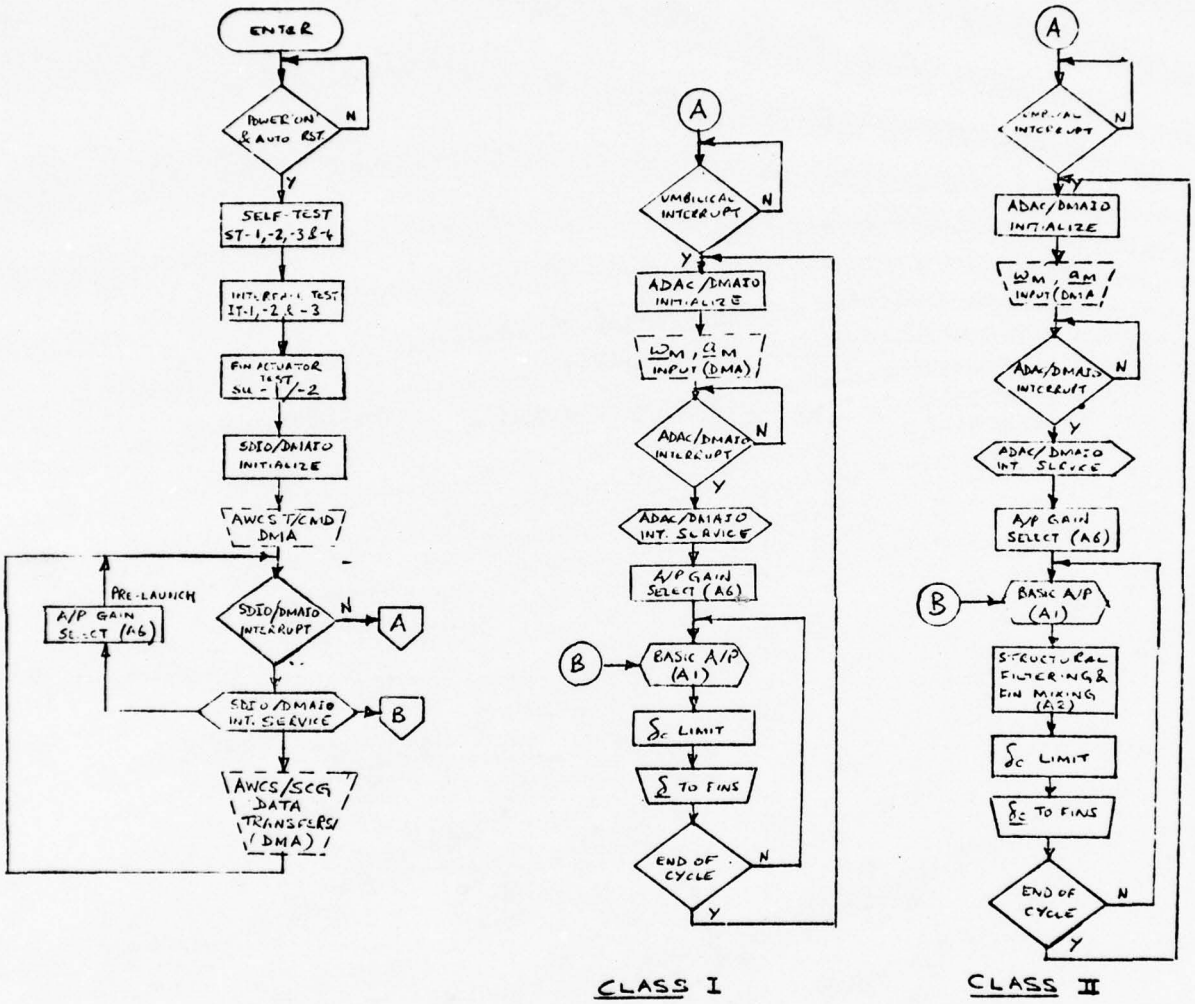


(B) CLASS II MISSILE AUTOPILOT

Figure 61 Autopilot Microcomputer Configurations, Block Diagrams

Microcomputer Operation

Flow charts, Figures 62 (A) and (B), show the overall functional operation of the two autopilot microcomputers.



(A) Pre-Launch (B) Post-Launch

Figure 62 Autopilot Subsystem Operational Flow Charts

Initialization and Self-Test - The initialization and self-test programs for the autopilot are virtually the same as those for the head control microcomputer with the exception of module SU-1, which is replaced by SU-2 for a Class II missile with four independent fin actuators (Ref. R-2).

AWCS - Umbilical Interface - This is electrically the same as all other microcomputers in the system, i.e., DMAIO/SDIO (1553A) and umbilical interrupt. Following the transmission of subsystem test results to the AWCS upon command, the autopilot microcomputer is initialized by the AWCS in a control band based upon Mach number and altitude before launch. The resulting occurrence of the SDIO/DMAIO interrupt and associated AWCS Receive command initiates the execution of the control gain selection program (A6).

Autopilot - Separation of the umbilical and the activation of the umbilical interrupt results in the ADAC/DMAIO interrupt being enabled to synchronize autopilot program execution to the ADAC interval timer. The latter timer initiates the simultaneous sampling of all analog sensor inputs ($\underline{\omega}_M$, $\underline{a}_M$) at 4 msec intervals for the Class I missile system and 2 msec intervals for the Class II system due to the wide bandwidth of the structural filters. Sensors are sampled synchronously with the gyro reference frequency at the peak amplitude, as in the case of the head control gyros. The analog samples are converted to 10-bit digital values for the Class I autopilot and 12-bits for Class II (Ref. R-2). Following the ADAC/DMAIO interrupt, processing proceeds one channel at a time with the outputting of each channel's fin command in sequence, in the case of the two-channel Class I autopilot. For the Class II system, outputs are made as mixed

channel pairs i.e. pitch and roll (fins 2 & 4) and roll and yaw (fins 1 & 3) respectively, to minimize the throughput required.

Autopilot 'g' commands, $\underline{a}c$, are received every 100 msec via the SDIO/DMAIO channel asynchronous to the cyclic 2 - 4 msec sensing of body rate gyros and accelerometers by the ADAC. Two 4/6-word buffer areas are assigned in the RAM module (corresponding to a 2 or 3-channel system), to enable the most recent 'g' data to be used at the commencement of each autopilot computing cycle. The inactive buffer is dedicated to the DMAIO storage of new 'g' commands should they be input during a current autopilot cycle.

Band selection as a function of missile speed (V_M) and elapsed time (t) since launch, is made by executing program module A6 every major sampling interval (4 msec, Class I; 2 msec, Class II). Although a 100 msec update interval would meet the Phase I requirements for bandswitching, the small program size (30 words) and computer load warrants the more simple timing template. Elapsed time is derived from the ADAC interval timer and associated interrupt.

Microprocessor Performance

The results of applying the 8 and 12-bit microprocessors to the autopilot function are shown in Table 22 for Class I and II missiles respectively.

TABLE 22

AUTOPILOT VS MICROPROCESSOR PERFORMANCE

FUNCTION	μCPU-1 (N-MOS) 8080A +SMPY	8080A +HMPY	Z80 +HMPY	6100 +HMPY	μCPU-2 (BIPOLAR) 8080 +SMPY	8080 +HMPY
MICROPROCESSOR CLOCK (nsec)	380	380	400	325	150	150
CLASS I MISSILE (PER CH.)						
No. of Clock Cycles		554	535	638	589	327
Computational Delay (μsec)		211	254	207	88	49
Total Program Size (Bytes)		497				
CLASS II MISSILE (PER CH.)						
No of Clock Cycles	-	1340	1325	1330	2317	877
Computational Delay (μsec)	-	509	530	432	348	132
Total Program Size (Bytes)	-	980				
CLASS II MISSILE (2 CHS.)						
No. of Clock Cycles	-	2426	2400	2373	4606	1593
Computational Delay (μsec)	-	922	960	771	691	239

Class I Autopilot - Based upon the acceptable computational delays determined in the Phase II study, the N-MOS 8080A microprocessor, using a software multiply subroutine, meets the performance requirements for a Class I autopilot. The basic 8080A uses 670 μsecs of the 800 μsec allowance for a single-channel, bandswitched autopilot. This is largely due to the small number of multiplies required i.e. 2. Memory requirements are also modest - 497 bytes program, making the 8080A a very cost effective processor for this application.

Class II Autopilot - For the higher performance, 3-channel, 4-fin Class II autopilot, incorporating structural filters and fin mixing, either the N-MOS 8080A with μbus hardware multiply module (HMPY-1) or the bipolar equivalent meets the 600 μsec computational delay requirement, depending on whether one or two control channels are computed during the latter interval.

Figure 63 illustrates the single-channel, low-throughput approach, where each rate gyro is sampled and the corresponding δ_C computed and mixed with the δ_C of a previously computed complementary channel e.g. pitch and roll.

The N-MOS 8080A with a HMPY-1 module requires approximately 500 μsec to execute a single control channel loop.

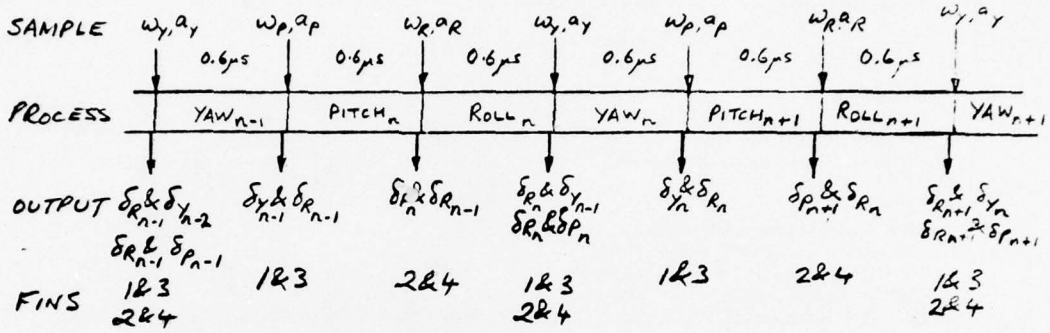
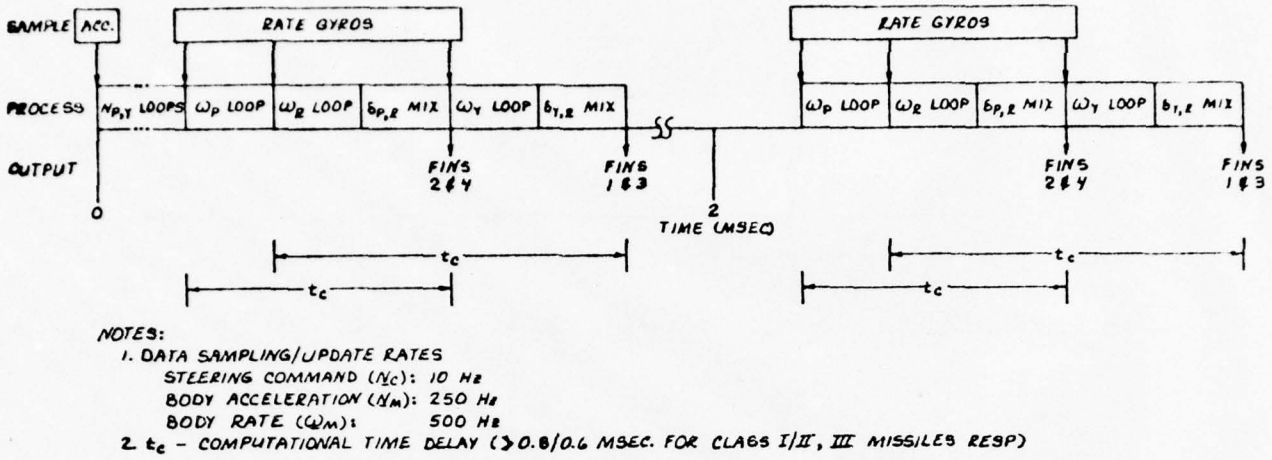


Figure 63 Class II Autopilot Computing Sequence - Single Channel Fin Updating

Figure 64 shows the autopilot timing diagram presented in the Phase II Report. This method requires over 900 μsec to process and mix two control channels with the same micro computer configuration, i.e. N-MOS 8080A & HMPY-1, thus forcing the use of the higher speed bipolar equivalent with attendant increases in cost, parts count and power consumption.

Compared to the simple Class I autopilot involving 2 multiplies per channel, the Class II system requires 11 multiplies for the pitch or yaw channel and 12 for roll, (9 multiplies for the structural filter in each channel). Memory requirements were approximately 1K bytes program.



NOTES:
1. DATA SAMPLING/UPDATE RATES
   STEERING COMMAND (N_C): 10 Hz
   BODY ACCELERATION (N_M): 250 Hz
   BODY RATE (ω_M): 500 Hz
2. t_c - COMPUTATIONAL TIME DELAY (>0.8/0.6 MSEC. FOR CLASS I/II, III MISSILES RESP)

Figure 64 Class II Autopilot Computing Sequence - Two Channel Fin Updating

Instructions Used - Table 23 lists the instructions exercised for each microprocessor type evaluated. The 8080A required 14 different types, the Z-80 11 types and the 6100 5 types.

Computing Precision - ADAC module A-D and D-A conversions were assumed to be 10 bits for Class I and 12 bits for the Class II missile. Computing precision was 16-bits, which enabled the Z-80 to accomplish the task with fewer clock cycles than the 8080A.

Subsystem Performance - Table 24 summarizes the subsystem I/O and computational time delays.

TABLE 23

AUTOPILOT - INSTRUCTIONS EXERCISED

INSTRUCTION CATEGORY	MICROPROCESSOR 8080A	Z80	6100
Data Transfer	LHLD	LD	DCA
	XCHG	EX	
	MVI	PUSH	
	MOV	POP	
	SHLD		
	PUSH		
	POP		
	STA		
Arithmetic & Logical	DAD	DAD	AND
	SUB	SUB	TAD
	SBB	INC	CLA
	XRA	SBA	CIA
Program Control	JP	CALL	
	CALL	JR	

NOTES

* Class II missile only.

TABLE 24

AUTOPILOT LOOP DELAYS

ELEMENT	DELAY (μSEC) CLASS I	CLASS II	COMMENTS
DMA/ADAC Input	46.8	70.2	Class I: 4 sensors Class II: 6 sensors
Autopilot Program	211.0	509	One channel using μCPU-1 + HMPY-1.
	-	(239)	(Two channels using μCPU-2 + HMPY-1)
DMA/ADAC Output	20.8	41.6	Class I: 2 fins
t_{AP} (μSEC)	278.6	620.8 (350.8)	

The performance of the N-MOS 8080A-based μCPU-1 with μBus HMPY-1 module is well within the 800 μsec Class I autopilot requirement, but only marginally satisfactory for the Class II subsystem. However, μCPU-2 with the CMOS/SOS or bipolar 8080 and HMPY-1 module provides a 75% performance margin for Class II autopilots.

6.1.3 Fuzing

The remaining major missile functions selected for applying an 8-bit microprocessor architecture was the fuze time-delay computation, in order to achieve high-lethality warhead detonation for any given end-game situation.

Microcomputer Configuration

Figure 65 is a block diagram of the applicable warhead fuzing microcomputer subsystem, which incorporates a medium speed μCPU-1 microprocessor and associated RAM/ROM-1, PDIO and SDIO/DMAIO modules. Interface with the other missile subsystems and AWCS is via the SDIO. The target detection pulse from the fuze receiver drives an interrupt to μCPU-1 and the subsequent detonation command to the warhead safing and arming device is output via the PDIO interface module. During initial flight tests a similar computer configuration, augmented with an ADAC/DMAIO module, would perform the telemetry function.

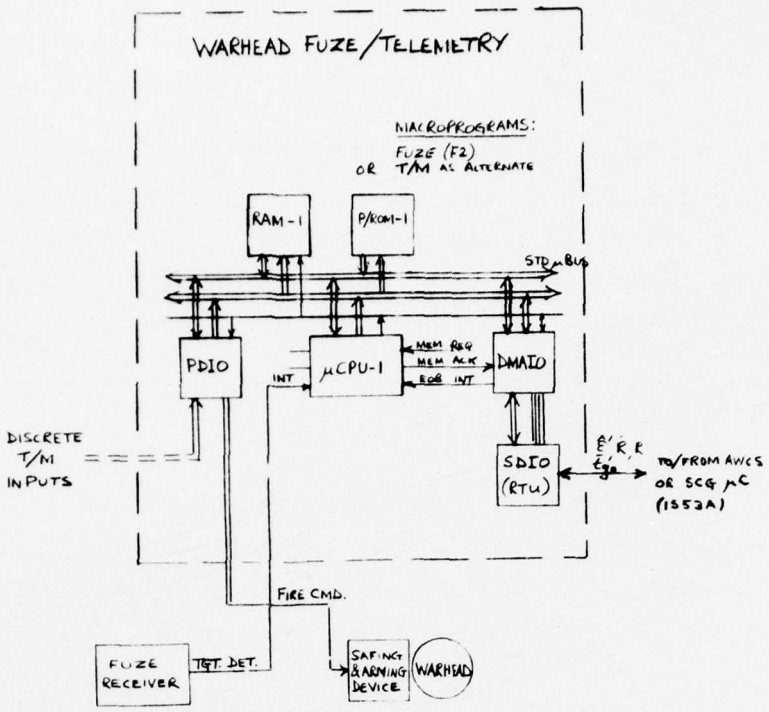


Figure 65 Fuzing Microcomputer Subsystem - Block Diagram.

Microcomputer Operation

The functional operation of the fuzing subsystem is shown in the flow chart of Figure 66.

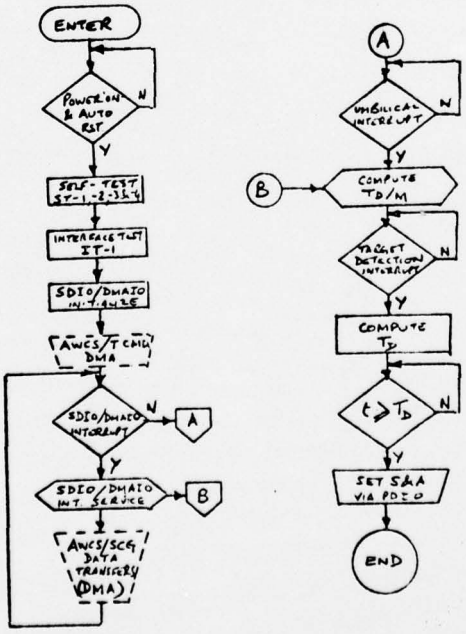


Figure 66. Fuzing Microcomputer Functional Flow Chart.

Initialization - The initialization sequence is the same as that described in the previous two microcomputer subsystems.

Self-Test - This program sequence incorporates all the microcomputer self-test programs (ST-1, -2, -3 and -4) used in the head control and autopilot computers and the digital interface test module IT-1. Upon completion of these self-test routines, the results are output to the AWCS upon command.

Time-Delay Computation - Separation of the umbilical from the missile results in the fuzing microcomputer being controlled by the seeker subsystem (SCGC) via the SDIO/DMAIO interface. The fuze microcomputer idles until an SDIO/DMAIO interrupt occurs and the warhead arming command is received from the SCGC and validated, together with missile-target relative velocity ($\dot{R}$), boresight error ($\underline{\varepsilon}$) and seeker gimbal angle (θ_S, ψ_S) data. These data are used to compute T_D/M, (see Figure 67). The target detection interrupt is also enabled at this time and the computer idles again until the fuze receiver detects the target and generates a target detection pulse, whereupon the time (T) between arming and detection/fuzing is computed, M is estimated, and hence the time delay T_D is calculated. The warhead is then detonated T_D seconds after initial target detection, via the PDIO and safing and arming device.

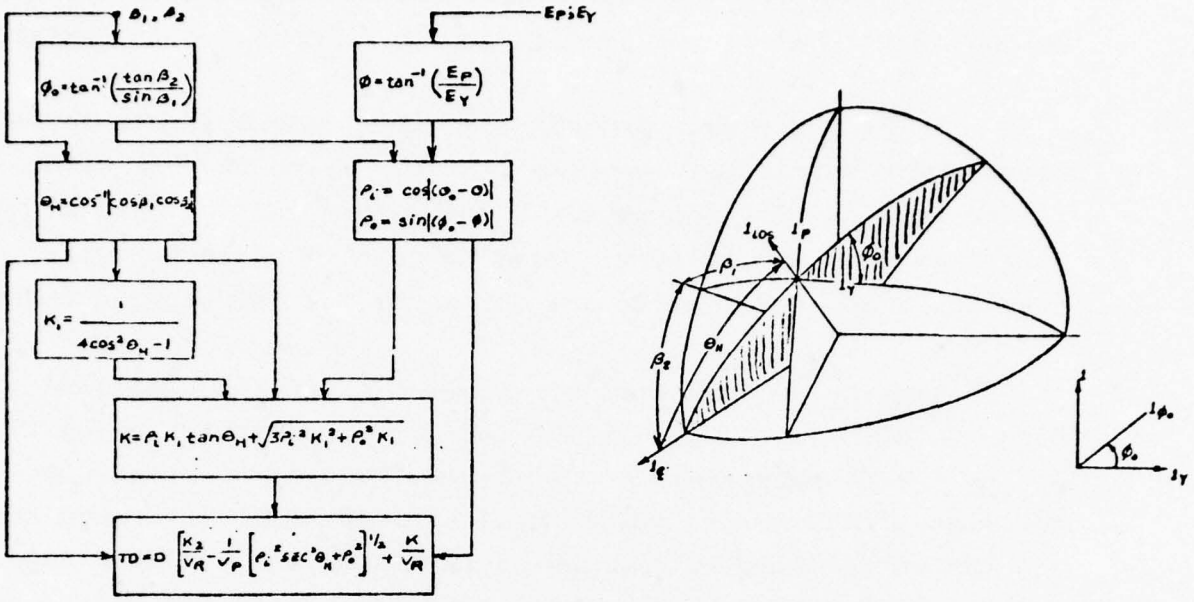


Figure 67. Time Delay Program Module F-2 Flow Chart

Microprocessor Performance - The performance of the 8080 only was evaluated for the fuze time delay function and the results obtained are summarized in Table 25 below.

Due to the large number and various types of arithmetic and trigonometrical operations required, (e.g., add, subtract, multiply, divide, square root, cosine), logarithm techniques were used to avoid round-off errors and the need for multiply and divide operations. Log, antilog and cosine values were stored in ROM.

TABLE 25

MICROPROCESSOR PERFORMANCE VS WARHEAD FUZING

PARAMETER	μCPU-1 8080	μCPU-2 Bipolar 8080
Machine Clock Period (nsec)	380	125
No. of Clock Cycles	16,447	16,447
Computational Delay (msec)	6.25	2.05
Program Memory Size (bytes)	1329	1329
Data Memory Size (bytes)	125	125

The computational delay times given in Table 25 represent the time taken to compute the optimum value of the detonation time delay (T_D) following receipt of the arming command. Approximately 90% of this time is used to compute T_D/M and the remainder of the time, i.e., 720 μsec approx for the 8080, is required to determine T_D after target detection. Hence the performance of the N-MOS 8080 would meet the requirements of intercept situations where the optimum values of T_D and T are not less than 0.72 and 5.6 msecs respectively.

Logarithmic Computations - Of the three computational approaches considered for the simple 8-bit byte processor, i.e., fixed-point software, fixed-point with hardware multiply and floating-point using logarithms, the logarithm technique offered the following advantages:

1. Multiplication and division reduced to add and subtract operations respectively.

2. Elimination of significant round-off errors in arithmetic sequences.

3. Simple to code and modify

4. Accomodates wide dynamic range operands without multiple byte operations.

The following paragraphs describe the conversion and manipulation of data using equivalent logarithm representations.

A subroutine is called to generate the equivalent logarithm (base 2) of each operand involved in multiplication, division and square root operations. This subroutine translates 11-bit fixed-point sign/magnitude numbers into corresponding 3-byte sign/characteristic/mantissa representations as shown in Figure 68.

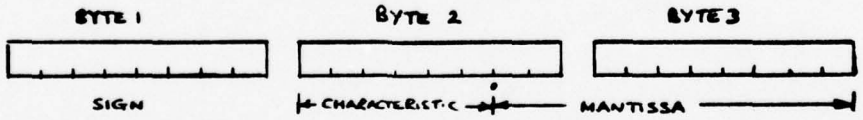


Figure 68
Logarithm Number Representation

The characteristic is determined by counting the number of shifts required to obtain one non-zero bit to the left of the binary point. The sign of the characteristic is determined by the direction of shifting required, i.e., shift right-positive, shift left-negative. The absolute value of the 10-bits magnitude is used for either direct or relative addressing of the corresponding mantissa stored in ROM. The following operations were coded using logarithms:

Operation	Executing
$A * B$	$\log_2 A + \log_2 B$
$A \div B$	$\log_2 A - \log_2 B$
$\sqrt{A}$	$1/2 \log_2 A$ (i.e., right shift.)
$A - 2^N$	$\log_2 A - N$.

Instructions Used - A total of 29 out of 78 available instruction types were exercised in coding the F2 module as listed in Table 26. Complete programs are given in Ref. R-23.

TABLE 26

FUZING - INSTRUCTIONS EXERCISED

(8080)

INSTRUCTION CATEGORY	MNEMONICS		
Data Transfer	LHLD	RAR	
	MOV	MVI	
	XCHG	PUSH	
	LXI	POP	
	STA		
	SHLD		
Arithmetic & Logical	ORA	XRA	ADC
	CMA	ANI	
	ADI	SUB	
	ACI	ADD	
	DAD	ANA	
	INX	SBI	
Program Control	JP	JZ	
	CALL	RET	
	JM	JMP	

AD-A042 466 RAYTHEON CO BEDFORD MASS MISSILE SYSTEMS DIV F/G 16/4.1
MODULAR DIGITAL MISSILE GUIDANCE PHASE III.(U)
MAY 77 F J LANGLEY
UNCLASSIFIED BR-9448 ONR-CR233-052-3 N00014-75-C-0549 NL

3 OF 3
AD
A042466

END
DATE FILMED
8-77
DDC

6.2 μCPU-3 & μCPU-4 Architectures

Table 27 lists the significant features of the two processor architectures evaluated for μCPU-3 & μCPU-4 applications, viz: PDP-11/34 and AN/UYK-20(V). Figure 69 shows the functional configuration of each machine.

Both machines are of the general-register class, and neither processor uses implied accumulators. The PDP-11/34 has 8 general-registers, two of which are normally assigned as a stack pointer and program counter respectively, leaving 6 as actual general-purpose registers. In contrast, the AN/UYK-20(V) has 16 truly "general" registers, with an additional program counter. By programming a status register, an alternative set of 16 registers can be addressed through the instruction set. In their basic form, both machines provide a fixed-point arithmetic capability and hence fit the μCPU-3 category. Both the PDP-11/34 and AN/UYK-20(V) *offer a firmware* floating-point arithmetic option, thereby meeting the requirements of μCPU-4 through modular growth. Double precision multiply and divide, together with square-root, trigonometric and hyperbolic instructions are other firmware options for the AN/UYK-20(V).

Bit, byte, word and double operand processing instructions are common features of both processors, with the AN/UYK-20(V) accomodating 4-bit literal operations in addition to the above.

The PDP-11/34 CPU is memory synchronous, to accommodate different memory cycle times, whereas the AN/UYK-20(V) CPU is designed to operate with a fixed memory cycle time of 750 nsecs.

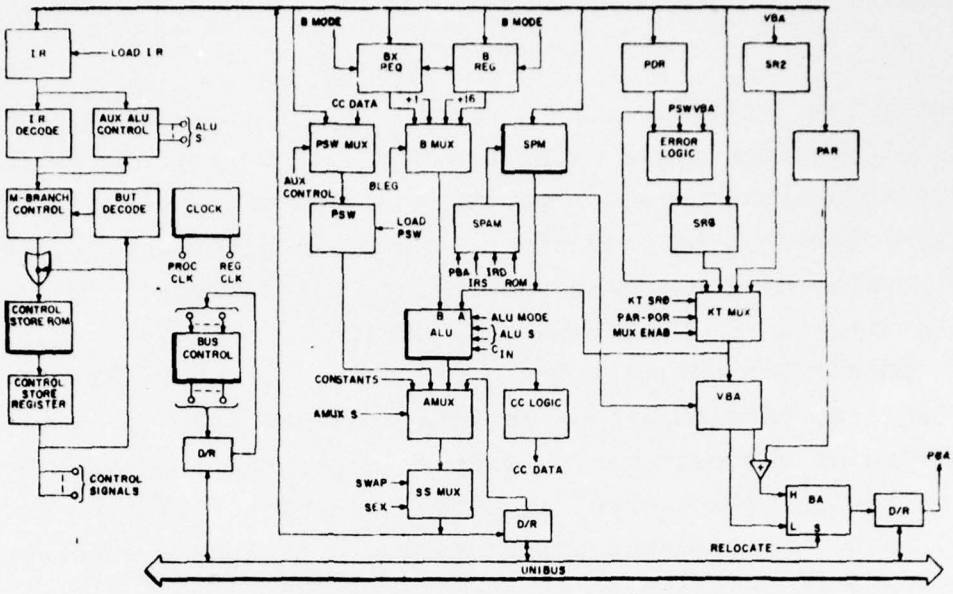
TABLE 27

CPU-3 & -4 TARGET MICROPROCESSOR ARCHITECTURAL FEATURES

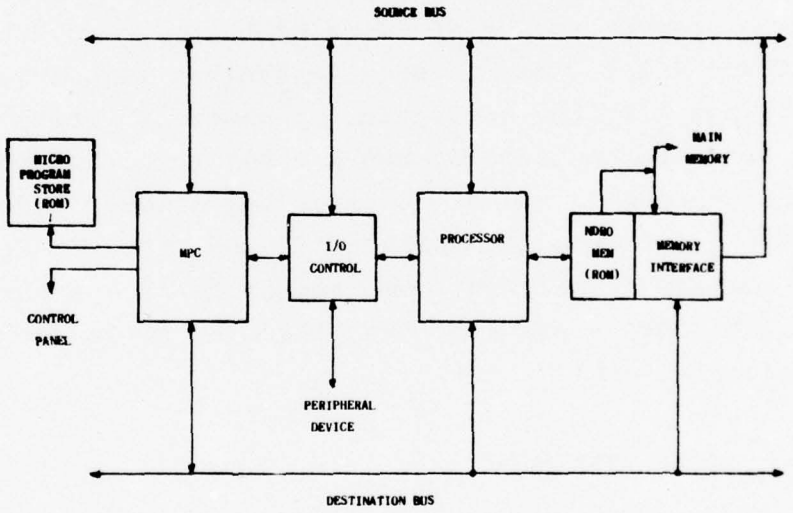
TARGET PROCESSOR	WORD SIZE (BITS) INSTRN.	DATA	REGISTERS GENERAL	DEDICATED	ADDRESSING MODES	INSTRN. SET	MEMORY CYCLE TIME (μsec)
PDP-11/34	16	16	6x16	PC(16) SP(16)	Immed (Inc/Dec) Direct (R-R) Indirect Indexed Relative	65 Fixed-point arithmetic, incl. mpy & div. Also medium-speed floating-point processor.	0.7 or 1.0
AN/UYK-20	16	16	16x16 plus alter-nate 16 x 16	PC (16) SR (16) 2 ea. PR (16) 64 ea. BR CR (32) MR	Immed (Inc/Dec) Direct (R-R) Indirect Indexed Relative	Fixed-point arithmetic incl. mpy & div. Firmware floating point as option.	0.75 (fixed)

Legend:

ACC - Accumulator
PC - Program counter
SP - Stack pointer
SR - Status Register
IR - Index Register
PR - Page Register
BR - Breakpoint Register
CR - Real-time Clock Register
MR - Monitor Clock Register



(A) Digital Equipment Corp. PDP-11/34



(B) Navy AN/UYK-20(V)

Figure 69 Target Microprocessor Architectures, μCPU-3 & μCPU-4, Block Diagrams

However, the latter features, together with the respective memory, machine and clock cycle times are, to a large extent, technology rather than architecture dependent. Hence, the number of memory and machine/clock cycles required to execute a program provides an ideal basis for subsequent performance evaluations. This allows flexibility in the selection of a device technology, e.g., N-MOS/bipolar/CMOS-SOS, and integration level, (LSI/VLSI) to match the required throughput for missile applications. Architecture then remains the determining factor in programming efficiency, the effectiveness of any given architecture being reflected in the total number of instructions required to perform a function and the proportion of overhead operations involved.

6.2.1 Steering Command Generation

With respect to the SA-CW ad SA-PD radar seekers identified for the Class I and Class II missile systems respectively, (see Figures 10 and 11), the generation of boresight error and 'g' commands for missile guidance and control requires the initial progression through a number of radar operational modes culminating in target acquisition, as described in the Phase II Report. The steering command generation loop becomes fully operational in the Target Track Mode, exercising the following series of functions/operations:

1. Radar Data Input
2. Burst Weighting
3. Corner Turning
4. Spectrum Analysis

5. Target Selection

6. State Estimation

7. Guidance Law

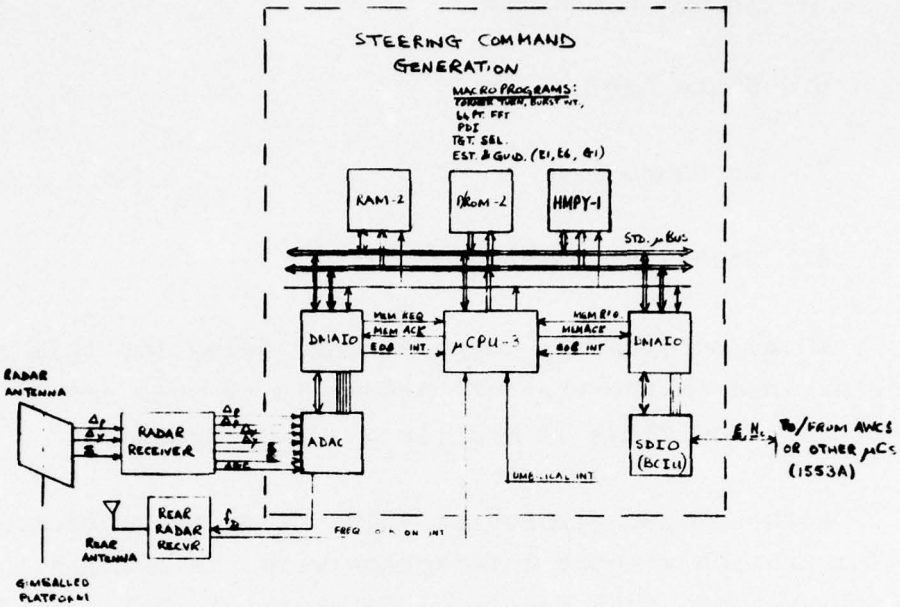
8. Steering Command Output

Total permissable computational delay for this forward loop, as determined in the Phase II study, is 40 msec and 30 msec max. for Class I and Class II missile guidance systems.

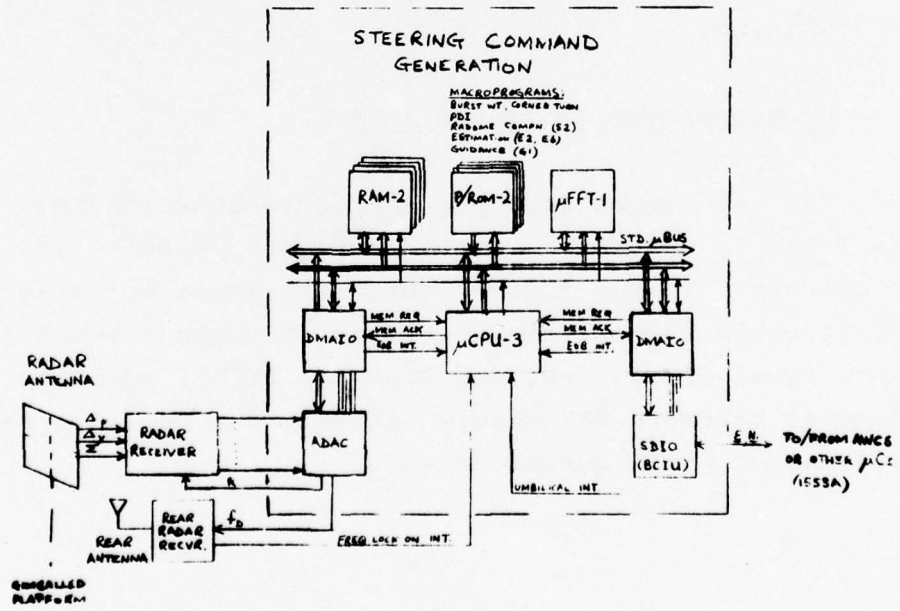
Although target/clutter acquisition is performed as an "open loop" operation without interaction with the missile and platform control systems, post detection integration (PDI) effectively replaces the estimation and guidance functions in the SCG group, and therefore warrants evaluation as a potential general-purpose computer function.

Microcomputer Configurations

The two target computer configurations considered for the Class I and II system are shown in Figure 70 (A) & (B) respectively. It was assumed that a hardware multiply module (HMPY-1) would provide the necessary throughput boost for SA-CW seeker signal processing, and that the HMPY-1 would be replaced by a 64-point hardware FFT module (μFFT-1) for multiple range-gate/FFT processing of SA-PD sensor data.



(A) Class I SA-CW Radar Seeker



(B) Class II SA-PD Radar Seeker

Figure 70 Steering Command Generation Microcomputer Subsystem Block Diagrams

Substitution of μCPU-3 with μCPU-4 to obtain the floating point arithmetic capability would be advantageous for Class II max. missiles employing variable gain filters and program module E3.

The Steering Command Generation microcomputer system is in effect the master/executive computer in the federated system after launch. The SDIO module becomes the bus control interface module (BCIU) of the 1553A serial digital interface system. Commands are generated by SCG computer program to address each of the other three microcomputers (head control, autopilot and fuze) via their respective SDIO modules. Through the 1553A command/response mechanism, the head control microcomputer can be commanded by the SCG microcomputer to receive boresight error commands, and the autopliot microcomputer 'g' commands, when these data are generated, i.e., every 100 msec.

Interface with the radar receiver is via the ADAC/DMAIO module pair. Seven input channels for the SA-CW seeker, Σ, Δp, Δy (complex) and AGC, and eighteen channels for the SA-PD seeker (5 range bins for the acquisition mode). Outputs are estimates of target Doppler ($\hat{f}_{TGT}$) for the CW receiver complemented with target range ($\hat{R}_{TGT}$) for the Class II pulse radar receiver.

In the latter case, sampling of receiver video outputs for A-D conversion is synchronized to the range gating waveform.

Microcomputer Operation

Figure 71 is a first-level flow chart showing the functional operation of both the Class I (SA-CW) and Class II (SA-PD) SCG microcomputers. The chart is essentially the executive control

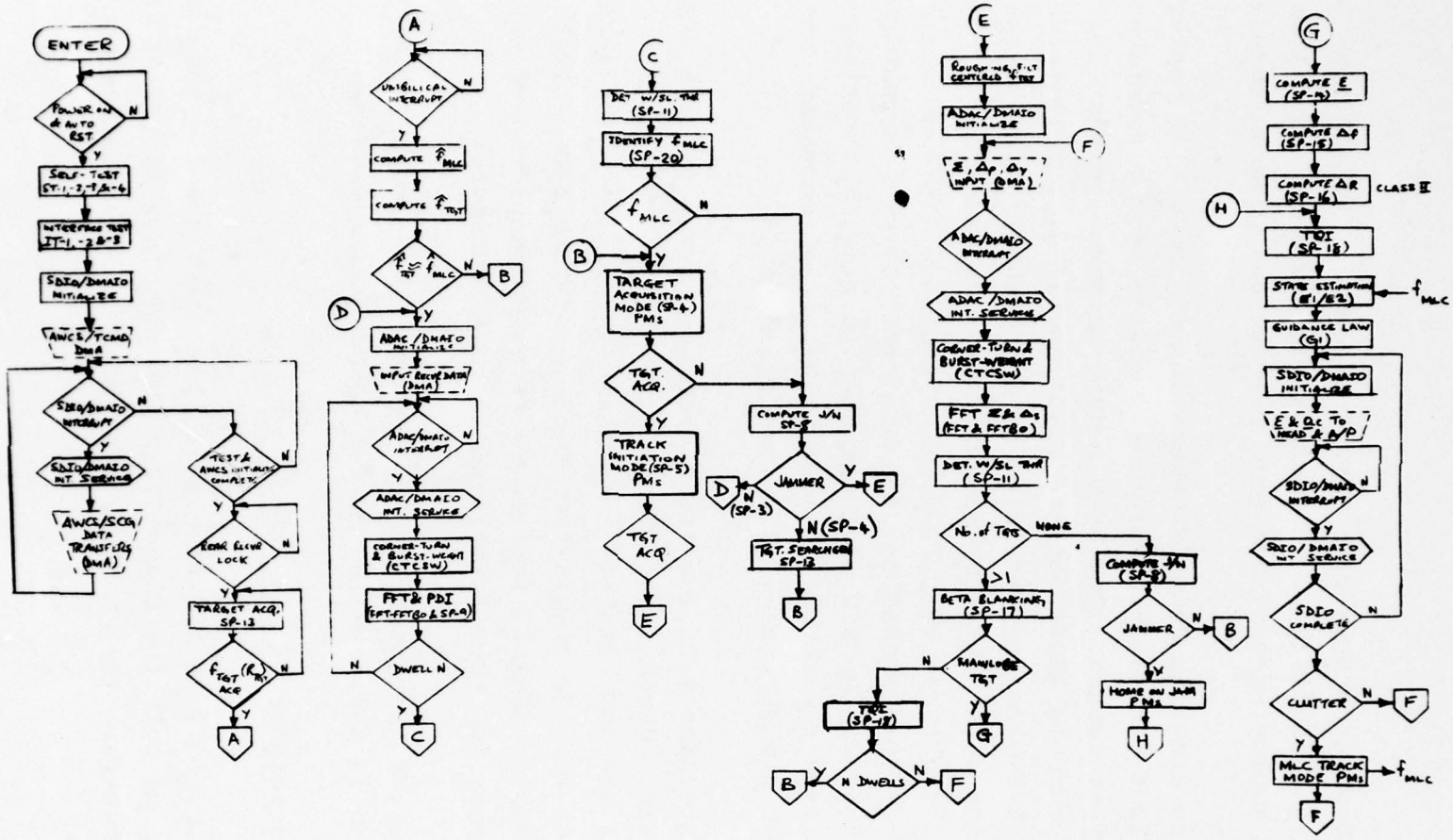


Figure 71 Steering Command Generation Subsystem, Operational Flow Chart.

program described in the Phase II Report, with self-test, AWCS intialization, estimation and guidance programs added.

Initialization and Self-Test - The SCG microcomputer is initialized in the same manner as the other missile subsystem computers following application of DC power to all macromodules.

Self-test program modules ST-1, -2, -3, & -4 are executed first, to verify proper operation of the μCPU, RAM, ROM, HMPY and/or μFFT modules. Satisfactory completion of these tests is followed by the checkout of its ADAC/DMAIO and PDIO modules by executing interface module test programs IT-1, -2 & -3. The SDIO/DMAIO interface is subsequently verified by an error-free response (reflected in the 1553A Status word), to AWCS commands via the umbilical.

SCG subsystem tests are performed through radio frequency (RF) and umbilical interfaces with the AWCS, as described in the following paragraphs.

AWCS - Umbilical Interface - Before launch, i.e. before an umbilical disconnect interrupt, the SCG microcomputer responds to AWCS commands in the same way as the other microcomputers in the system. The SDIO is effectively a remote terminal unit (RTU) in that no commands are generated by the microcomputer and the AWCS computer performs the master control function. After launch the SCG SDIO functions as a 1553A bus control interface unit (BCIU) with the SCG microcomputer assuming the role of the master computer in the system.

Since the radar seeker, estimation and guidance functions performed in Class I (SA-CW) and Class II (SA-PD) missiles use predominantly fixed parameters or pre-computed table look-up data, few data words are transferred from the AWCS to the SCG microcomputer before launch. However target velocity (and range for PD sensors) is trasmitted, to complement the angle data output as head aim commands to the Head Control microcomputer, also the mainlobe clutter Doppler frequency and range or time-to-go, the latter for Terminal Phase initiation. ECCM data is also a possibility in situations where the processing capabilities of the aircraft avionics system computer can complement the limited ECCM capability of the missile.

Although not specifically an AWCS-umbilical interface, prior to launch, the rear receiver of the missile receives target illumination radio frequency (RF) energy from the aircraft radar via a dedicated antenna. The rear receiver uses this RF signal as a reference to coherently set the target seeker local oscillator.

After frequency lock-on the rear receiver activates an interrupt line to the microprocessor which responds by executing signal processing modules to acquire the target Doppler frequency

(f_{tgt}) from a signal derived from the launch aircraft radar subsystem and injected into the front receiver of the missile. With the target Doppler determined, the Doppler tracking loop is closed by outputting f_{tgt}, via the D-A section of the ADAC module, to the rear receiver. In the case of the SA-PD seeker, target range (supplied via the 1553A interface) is used to select the appropriate range gate of the front receiver. Externally mounted missiles are then initialized to acquire the target in Doppler (and range for PD radar sensors) before launch.

The results of the self-test programs, rear receiver lock status, target Doppler and range acquisition status are transmitted to the AWCS in response to a 1553A Transmit Command.

Satisfactory operation of all missile microcomputer subsystems is followed by missile launch, separation of the umbilical, and automatic activation of the umbilical interrupt to the SCG microprocessor.

Launch Mode - The occurrence of an umbilical interrupt results in the transfer of control to the Launch Mode Supervisor. Mainlobe clutter (MLC) and target Doppler frequencies are predicted for the purpose of initializing subsequent Clutter and Target Acquisition Modes. Also, based on these predicted Dopplers, a decision is made by the executive to bypass the clutter acquisition phase if the Doppler separation between the MLC and the target is large enough.

Midcourse Mode - In the Class II case where a Midcourse Mode is employed, the Executive continues to designate the Launch Mode Supervisor to update the Doppler predictions until completion

of midcourse is indicated by range-to-go (R_{TGT}) or time-to-go being less than the value stored in RAM before launch.

Clutter Acquisition Mode - In cases of wide separation between the main lobe clutter and target Doppler frequencies (f_{MLC} and f_{tgt} respectively) e.g. approaching targets, the Clutter Acquisition Mode is skipped and the Target Acquisition Mode is initiated by the Executive program.

If, on the other hand, the latter two Doppler frequencies are narrowly separated, as in low-altitude, tail-chase, engagements, the Executive program designates the Clutter Acquisition Mode Supervisor which proceeds to call a series of program modules to determine the true mainlobe clutter frequency as opposed to the currently predicted value. This process involves ten, consecutive, 5 msec radar dwell periods, with the Fast Fourier Transform (FFT) and post detection integration (PDI) algorithms, (U-21 and SP-9), executed once in each dwell for the Class I SA-CW seeker, and five times per dwell, (once per range bin), for the Class II SA-PD seeker. The monopulse sum (Σ) channel only is processed in both cases.

In the Class I configuration, the ADAC/DMAIO modules input the in-phase and quadrature components of the sum channel to the RAM module at 5 msec intervals, using the pre-programmed ADAC interval timer. For Class II systems, ADAC input sampling is synchronized to the front receiver range gate generator which in turn uses the rear receiver to determine the effective "transmit" pulse time, (i.e. time of receipt of the aircraft radar pulse), and the appropriate time intervals for the range gating pulses/strobes.

After 10 dwells the results of the FFT/PDI processing are used in a thresholding process, (program modules SP-10/-11). Filter bin values exceeding a predetermined threshold magnitude are summed on a filter by filter basis, and on an individual range basis for the PD Class II system. f_{MLC} is found by executing a sort routine, (program module SP-20), to determine the highest thresholded value and associated filter bin.

If the mainlobe clutter Doppler frequency is successfully identified, a "Clutter Acquisition Complete" flag is set and the Executive responds by calling the Target Acquisition Mode Supervisor. If mainlobe clutter is not identified, the Jammer-to-Noise ratio is computed (program module SP-8), to determine if the missile is being jammed thus obscuring main-lobe clutter. If the missile is being jammed a flag is set, the Executive then transfers control to the Track Mode Supervisor, and target acquisition is bypassed. If the missile is not being jammed, clutter acquisition is again attempted and if after a specified number of attempts, mainlobe clutter has not been acquired (no flag set), the Executive calls the Target Acquisition Mode Supervisor.

Target Acquisition Mode - The initial processing sequence determines the initial position of the acquisition roughing filter(s). If acquisition must take place in a clutter environment, the acquisition roughing filter is positioned such that its low frequency corner is approximately 1 KHz above the mainlobe clutter Doppler for approaching targets and its high corner 1 KHz below the mainlobe clutter Doppler for receding targets. For approaching targets in a non-clutter environment, the high frequency corner of the roughing filter is positioned at the upper edge of the target Doppler uncertainty region.

Once the initial roughing filter position is determined, subsequent roughing filter positions and range-gate positions are determined by the target range/Dopppler search generator program module SP-13. This program generates a sequence of range gate and roughing filter positions such that the target range and Doppler ambiguity region (designated as one search "Frame") is covered in a cyclic manner.

For a specified range and Doppler, a target acquisition sequence is performed that is identical to that performed to acquire mainlobe clutter. The only difference is that after detection with the sliding window threshold, the detected signals are not summed in range. If there has been one or more targets detected for this range-Doppler position, the "target acquisition complete" flag is set and the Executive proceeds to call up the Track Initiation Mode Supervisor. If there were not target detections, the Jammer-to-Noise ratio is computed. If it is determined that the missile is being jammed, the Executive immediately designates the Track Mode Supervisor where Home-On-Jam tracking will be initiated. If the missile is not being jammed, the supervisor calls SP-8 which generates the range and Doppler positions for the next part of the Range/Doppler ambiguity region. This process continues until target acquisition is accomplished.

Track Initiation Mode - The Track Initiation Mode performs the function of acquisition verification by reacquiring the target in a narrower-band roughing filter and a narrower Doppler cell (i.e., the track "dwell" of 20 to 40 msec as opposed to 5 msec). Estimates of range and Doppler errors are also computed to initialize the range and Doppler tracking filters.

The first operation performed is the centering of the narrow-band track mode roughing filter and the track mode range gate on the range and Doppler "coordinates" determined in target acquisition. The spectrum of the roughing filter is then analyzed by executing a 64-point FFT on a single dwell. Target detection is then accomplished with the sliding threshold program (SP-11). A minimum of three range channels are involved in this process for the Class II system and one channel for the SA-CW missile: the sum channel main tracking gate and the leading and lagging split gates both of which are used to determine range error. If target acquisition is accomplished, the range and Doppler tracking errors are computed and a "track initiation complete" flag is set. If the target is not acquired, the Jammer-to-Noise ratio is computed (program module SP-8). For ratio values indicating jamming conditions the Executive designates the Target Track Mode Supervisor (for HOJ Tracking). In the absence of active jammers, reacquisition is attempted for a specified number of times and then control reverts back to the Target Acquisition Mode Supervisor.

Target Track Mode - This mode commences with the centering of the track roughing filters on the predicted target doppler ($\hat{f}_{TGT}$) and the track range gates on the predicted target range ($\hat{R}_{TGT}$). Both these estimates are outputs from the estimator modules (E1 or E2). Data is then collected for a single dwell (typically 20 to 40 msec as opposed to 5 msec in the acquisition mode). For the SA-CW radar sensor, 3 channels, (Σ,Δ_p, Δ_y), are sampled by the ADAC and input to RAM via the DMAIO. For the PD radar sensor, two additional channels are sampled, (Σ+ and Σ-), to determine range error using a split range gate on the monopulse sum channel. The 64-point FFT algorithm is then performed on each channel followed by the sliding threshold detection algorithm (SP-11) on the sum channel main tracking gate to determine

if any skin track targets are present. If at least one target is present track processing continues in the skin-track mode. To determine if any of the detected targets lie outside the seeker antenna mainlobe, the Beta Blanking program module (SP-17) is executed. For all targets passing the Beta blanking check, i.e. mainlobe targets, the pitch and yaw boresight errors and the range and Doppler tracking errors are computed, using program modules SP-14, SP-15 and SP-16 respectively. The absence of targets detected in the mainlobe results in a flag being set by the Supervisor program, causing the Executive to transfer control back to the Target Acquisition Mode Supervisor.

Mainlobe target processing continues with the execution of the Track Quality Indicator (TQI) program module (SP-18) to determine the signal-to-noise ratio for each error, which is inversely proportional to the variance on the measured errors.

The Target Selection program (SP-21) is executed immediately after the TQI function to select a single target when multiple targets are present. SP-21 also incorporates the logic necessary to handle range and/or Doppler gate stealer types of deceptive ECM.

If no targets were detected in the output of the Σ channel main tracking gate, the Σ channel Jammer-to-Noise (J/N) ratio is computed to determine if the seeker is being jammed. If the seeker is not being jammed, a missed look flag is set and sensed by the TQI (SP-18) routine, (i.e. no data is available on this dwell). After a specified number of missed looks in a row, the Executive redesignates the Target Acquisition Supervisor.

If jamming is detected, the Home-On-Jam (HOJ) flag is set

and sensed by the TQI program module. For high jammer-to-noise ratios, the HOJ boresight errors are computed from a single FFT cell. This allows the radar sensor to take advantage of the difference in jammer spectrums to possibly obtain guidance data on a single jammer in a multiple jammer formation. If the J/N ratio is not adequate for this purpose, the boresight errors are computed for all FFT cells and averaged to obtain a single pitch and yaw error estimate. (The HOJ angle error algorithm is the same as that shown for skin track). Beta blanking is used to eliminate jammers in the sidelobes. If the jamming target passes the beta blanking check, the Radial Angle Gate program module (SP-19) is called. The purpose of the radial angle gate logic is to enable the early resolution of a single blinking jammer in a multiple blinking jammer environment. After passing these checks SP-18 is computed and the data used as inputs to the estimation routine. In cases where the HOJ boresight error is computed for only a single FFT cell, if either beta blanking or radial angle gate logic resets it, the multiple cell boresight error is computed to obtain useable guidance information.

Main-Lobe Clutter Track Mode - The Main Lobe Clutter Track Mode is called to track the mainlobe clutter Doppler when a target engagement is taking place under ground clutter conditions. The processing sequence is the same as the Clutter Acquisition Mode except that if the mainlobe clutter is acquired, the mainlobe clutter tracking error, Δf_{MLC}, and TQI are computed as inputs to the estimator module.

Microprocessor Performance

All program modules associated with the Target Track Mode were coded and compiled using the PDP-11/34 instruction set and the FORTRAN IV PLUS compiler of the Digital Equipment Corporation. The compiler was resident in a PDP-11/70 computer at the company's nearby Maynard facility. Approximately 4,000 lines of object code were compiled. In addition, the Angle Track Error program module (SP-14) was selected as an effective benchmark program and re-coded in assembly language to provide an indication of the compiler's efficiency.

To assess the performance of the AN/UYK-20(V) architecture and instruction set, the SP-14 module was coded both in CMS-2 and assembly language (ULTRA 16-2). The Navy SPL/1 higher-order-language was also considered for the AN/UYK-20(V), but Release 4, the most effective version for missile guidance and control functions, was under development during this phase of the study.

μCPU-3 & μCPU-4 - Table 28 shows the results obtained with the PDP-11 FORTRAN IV PLUS code for the Target Track Mode SCG forward loop. Since the compiler exercised both fixed and floating-point instructions, the resulting code is compatible with a μCPU-4 type microprocessor module. Further, to be compatible with the medium-speed, firmware floating-point capability of μCPU-4 modules, (as well as firmware multiply and divide), the instruction execution times and corresponding numbers of memory cycles applicable to a PDP-11/34 with medium-speed FP11-A floating-point processor were used to arrive at the computational delays given in Table 28, (Ref. R-22). The PDP-11/34 offers either a 1 μsec core or 0.7 μsec MOS memory, the latter being equivalent to a RAM/(P)ROM-1 module. To provide throughputs which would be

TABLE 28

STEERING COMMAND GENERATION (TARGET TRACK MODE) VS MICROPROCESSOR PERFORMANCE
(PDP-11/34-BASED USING FORTRAN IV PLUS)

	CPU-4			
PARAMETER	RAM-1	RAM-2	RAM-2 HMPY-1	RAM-2 & FFT-1
MEMORY CYCLE TIME (μsec)	0.7	0.2	0.2	0.2
CLASS I (SA-CW)				
Memory Cycles:	100,471	100,471	No signifi-	8,659
Computational delay (msec):	185	135 *(46)	cant per-formance improvement.	19.5
CLASS II (SA-PD)				
Memory Cycles:	—	—	—	10,921
Computational delay (msec):	—	—	—	28.3 *(8.6)
PROGRAM SIZE (WDS):				1896

Legend:

* Approximate delay for PDP-11/45

consistent with higher speed, bipolar/CMOS/SOS RAM/(P)ROM-2 memory modules, a 0.2 μsec memory cycle time is used as an alternate to the slower MOS speed.

It should be noted however, that the ratio of CPU to memory time is 1.6:1 for the PDP-11/34 with 700 nsec MOS memory. The PDP-11/45 with bipolar RAM is approximately 4 times as fast as the PDP-11/34, which suggests a greater speed improvement than indicated in Table 28.

The compiler service was provided by the Digital Equipment Corporation gratis for the ONR study.

The path exercised through the Target Track Mode program modules was consistent with a single, mainlobe target in a clutter-free, non-ECM environment. Target detection was assumed to be in the primary Doppler tracking cell.

Floating-point arithmetic instructions were used by the FORTRAN IV PLUS compiler for all program modules except: the executive, mode supervisor, FFT and those programs which operate on the FFT matrix. The $\cos^2$ burst weighting module was also coded in floating-point for coding expediency, to the the detriment of run time. This module should be recoded in fixed point as an optimising procedure. Also, a pseudo computation of closing velocity was made, to approximate to the run time of a valid algorithm, by performing a square root on the sum of the squares of the line-of-sight rate. This also requires re-work in subsequent applications.

In summary, the results show that with non-optimised code written in FORTRAN IV PLUS, the PDP-11/ 34-type μCPU-4 with a firmware multiply, bipolar semiconductor memory and μFFT-1 module, meets the performance requirements of the steering loop for both Class I and Class II missiles. The effectiveness of the HMPY-1 module as a throughput booster for FFT computations in Class I missiles was hampered by the relatively slow register-to-memory and memory-to-memory move times of the PDP-11/34, (2 msecs and 2.5 msecs resp. approx. using RAM-2). Storing the FFT coefficients within the HMPY-1 module, i.e., in a ROM, together with faster load and store instructions, would better support this minimum hardware alternative to the μFFT-1 module for SA-CW radar missiles.

TABLE 29

POST DETECTION INTEGRATION VS MICROPROCESSOR PERFORMANCE
(ONE RADAR DWELL, PDP-11/34-BASED, USING FORTRAN IV PLUS)

PARAMETER	μCPU-4 + RAM-1	μCPU-4 + RAM-2
MEMORY CYCLE TIME (μsec)	0.7	0.2
CLASS I (SA-CW)		
Memory cycles:	6,912	6,912
Computational Delay (msec):	11.2	7.75 *(2.8)
CLASS II (SA-PD 5 Range Bins)		
Memory cycles:	34,560	34,560
Computational delay (msec):	56	38.75
PROGRAM SIZE (WDS):	197	197

Legend:

* Approximate delay for PDP-11/45

No multiplies are required for PDI computation hence the HMPY module has no significance. For the processing of one set of 64 complex spectral components, (Σ channel only) in Class I SA-CW seekers, the bipolar/CMOS/SOS semiconductor version of the PDP-11/34 could provide adequate throughput to support a 5 msec dwell period. However, for Class II SA-PD seekers, with five times the processing load, i.e. 5 range gates, the processing time exceeds the dwell time, therefore the PDI function should be incorporated in a hardware μBus module, e.g. the μFFT module.

To determine the compiler efficiency, the benchmark Angle Track Error program module, (SP-14), was re-coded is assembly language, again assuming a non-jamming environment and target detection in one Doppler cell. Using the PDP-11 MACRO assembler, program size was reduced by 26.5%, i.e. 175 wds. vs 238 wds, compared to the original FORTRAN program. A run-time reduction of 13% was achieved using the MACRO assembler code.

AN/UYK-20(V) Performance - To limit coding expense, only the benchmark program (SP-14) was re-coded in the Navy CMS-2M higher-order-language and ULTRA/16-2 assembly language for the AN/UYK-20(V) computer. The programs were compiled by Interteck Service Center of Huntsville, Alabama, using a Univac 1108 host computer with CMS-2M cross compiler and ULTRA/16-2 Cross assembler.

The results are shown in Table 30.

TABLE 30

SCG BENCH-MARK PROGRAM (SP-14) VS AN/UYK-20(V) PERFORMANCE

PARAMETER	CMS-2M	ULTRA/16-2
MEMORY CYCLE TIME (μsec)	0.75	0.75
MEMORY CYCLES	932	613
RUN TIME (μsec)	699	460
PROGRAM SIZE (WDS)	400	191

The ULTRA assembler reduced the program size by 52.25% and the run time by 34.2%.

PDP-11 VS AN/UYK-20(V) - Compared to the PDP-11 coding results, the AN/UYK20(V) ULTRA assembler produced a 191-word program compared to the 175-word PDP-11 MACRO program, an 8% reduction. On a compiler basis, CMS-2M produced 400 words and FORTRAN IV PLUS 238 words, a 40% reduction.

An examination of the code generated by the two compilers indicates that the CMS-2M compiler does little, if any, optimization while the FORTRAN IV PLUS compiler does considerable optimization, i.e. it performs: constant folding, elimination of redundant expressions, removal of invariant expressions from loops, and general-register assignment quite effectively. However, at least one function (ISHFT), which could easily be generated in-

line, is implemented via an external function. Since this function is heavily used on the SCG task, the overall FORTRAN IV PLUS compiler efficiency is considered to be approximately 75%.

FFT Routines - The 64-point FFT program was structured with the following two software modules to facilitate replacement by hardware modules.

1. **FFT** - Performs a 64-point fast Fourier transform on one channel of data and includes the logic to set up and call the FFT butterfly operation (FFTBO) algorithm 192 times.

2. **FFTBO** Performs a 2-point FFT on data furnished by the FFT module. Both programs are coded in FORTRAN, (for the PDP-11), thus forcing a less than optimum indexing scheme. FFTBO operates on a 64-point "common" buffer using a constant data base of 32 complex multipliers. The 64-point buffer is filled by the $\cos^2$ weighting program which calls FFT. The arguments to FFTBO are also in "common" as are the index values of the two points and that of the multiplier. These three indexes are computed by FFT for each call to FFTBO. This arrangement is fairly efficient for FORTRAN, but it has the disadvantage that FFTBO recomputes the index register values corresponding to the argument addresses each time it is called. If the subprograms were coded in assembler code, the arguments could be actual addresses passed via the index registers. Approximately 20% of FFTBO execution time would be saved.

PDP-11 Instructions Used - Table 31 lists the PDP-11 instructions exercised in the SCG Target Track Mode program modules for the assumed engagement scenario.

Of the 65 available PDP-11 instruction types, only 39 (60 %), of these were required to execute the target seeker functions.

Subsystem Performance - For system performance evaluation purposes, the steering command generation loop delays are summarized in Table 32. The time taken to generate new steering commands ($\underline{\varepsilon}$, $\underline{a}_c$), and transfer these via the 1553A serial digital interfaces to gimballed platform and autopilot microcomputer RAMs respectively, falls well within the maximum allowable values for t_{GUID} (Section 3).

TABLE 31

STEERING COMMAND GENERATION (SCG) - INSTRUCTIONS EXERCISED
(PDP-11/34 FORTRAN IV PLUS COMPILER)

INSTRUCTION CATEGORY	MNEMONICS					
Data Transfer	MOV					
	LDF					
	STF					
Arithmetic & Logical	ADD	CMP	TST	ADDF	CLRF	NEGF
	SUB	INC	SXT	SUBF	CLRB	TSTF
	MUL	ASL	CLR	MULF	CMPF	CFCC
	DIV	NEG	DIVF		LDCIF	STCFI
Program Control	BR	BLT	SETF			
	BNE	BGT	SETI			
	BEQ	BLE	BGE			
	BPL	JSR				
	BMI	RTS				

TABLE 32

STEERING COMMAND GENERATION LOOP DELAYS

ELEMENT	DELAY (msec)		COMMENTS
	CLASS I	CLASS II	
SCG (Track Mode) Program	19.5	28.3	
DMA/SDIO Interfaces	0.3	0.3	Total for $\underline{\epsilon}$ and $\underline{a}_c$
t_{GUID} (msec)	19.8	28.6	

REFERENCES

R-1 Hall, B.A. and Trainor, W.V., "Modular Digital Missile Guidance System Study" Ph. I Report, 30 June 1974, (DDC AD 784969)

R-2 Hall, B.A., and Langley, F.J., "Modular Digital Missile Guidance", Ph. II Report, 28 January 1976, (DDC AD B010399L)

R-3 Microprocessor Standardization Concepts Workshop, IEEE/NADC Warminster, Pennsylvania, 22-24 June 1976.

R-4 Microprocessor and Associated Large Scale Integrated Circuits (LSI) for High Reliability Applications Workshop NASA/JPL, Caltech., Pasadena, California, 27-29 October 1976.

R-5 Hall, B.A., Langley, F.J., Wefald, K.O., "Computer Design Requirements For Digital Air-to-Air Missiles", AlAA Guidance & Control Conf., San Diego, California, 16-18 August 1976.

R-6 "Military Handbook, Program Managers Guide for the Standard Electronic Modules Program", MIL-HDBK-246, 7 June 1976

R-7 "Military Standardization Handbook, Navy Standard Hardware Program, Application Handbook", MIL-HDBK-293 (NAVY) 5 September 1973

R-8 8080A Emulator (3000KT 8080 SK), Signetics Corporation, New Product Announcement No. 5, Internal Report, January, 1977.

R-9 9080 Emulator, An Example of a Microprogrammed Machine, Advanced Micro Devices, Inc., Internal Technical Report, March, 1977.

R-10 "Military Standard Aircraft Internal Time Division Command/Response Multiplex Data Bus," MIL-STD-1553A, 30 April 1975.

R-11 Springer, J., "Making Sense out of Delay Specs in Semiconductor Memories", Electronics, October 25, 1971.

R-12 Intel 8080 Microcomputer Systems User's Manual, Intel Corporation, September 1975.

R-13 Product Function Specification, Data Processing Set AN/UYK-20(V) and AN/UYK-20X(V), Sperry Univac Specification SB-10160 Rev. H, September 12, 1975

R-14 User's Handbook for AN/UYK-20(V) Computer, NAVELEX Specification PX 11079, Change No. 5, January, 1977.

R-15 Richardson, L.C., "PRIM Overview", Research Report, February 1974, USC/Information Sciences Institute Report ISI/RR-74-19, ARPA Contract DAHC 15 72 C 0308

R-16 Burke, E.L., Gasser, M., & Schiller, W.L., "Emulating a Honeywell 6180 Computer System" Technical Report, June 1974, Mitre Corporation Report MTR-2742, RADC Contract F19628-73-C-0001, RADC Report RADC-TR-74-137

R-17 Lee, Hseu-Jen. "CDL Simulation of a Microcomputer Based on INTEL-8080- Like CPU", Thesis for the Degree of Electrical Engineer in Computer Science, Northeastern University, June, 1975.

R-18 Microcomputer/Module Timing and Interface Simulation Software for ONR Modular Digital Missile Guidance Study, Phase III. (Available upon request to the Contracting Agency)

R-19 Zilog Z80-CPU Technical Manual, Zilog, Inc. 1976.

R-20 M6800 Microcomputer System Design Data Book, Motorola Inc., November, 1976

R-21 Harris HM-6100 CMOS 12 Bit Microprocessor Specifications, Harris Semiconductor, (Undated).

R-22 PDP-11 04/34/45/55 Processor Handbook 1976-77, Digital Equipment Corporation.

R-23 Microprocessor Performance Evaluation Software for ONR Modular Digital Missile Guidance Study, Phase III. (Available upon request to the Contracting Agency).

R-24 Cooley, J.W., & Tukey, J.W., "An Algorithm for the Machine Calculation of Complex Fourier Series", Math. Comp. Vol. 19, 1965 pp 297-301, MR 31 #2843.

APPENDIX A

OPERAND ADDRESSING MODES

DEFINITIONS

Immediate	Operand contained in the current instruction word. (Fig. A-1)
Direct/Absolute	Operand located in a register or main memory location designated by an absolute address contained in the current instruction word. (Fig. A-2)
Relative/Indexed	Operand located in main memory at the address obtained by adding/combining specific bits contained in the current instruction to the contents of a base/page register or the program counter. (Fig. A-3).
Indirect	Operand located in main memory at an address contained in a register or main memory accessed by an address contained in the current instruction. (Fig. A-4)

OPERAND ADDRESSING MODES

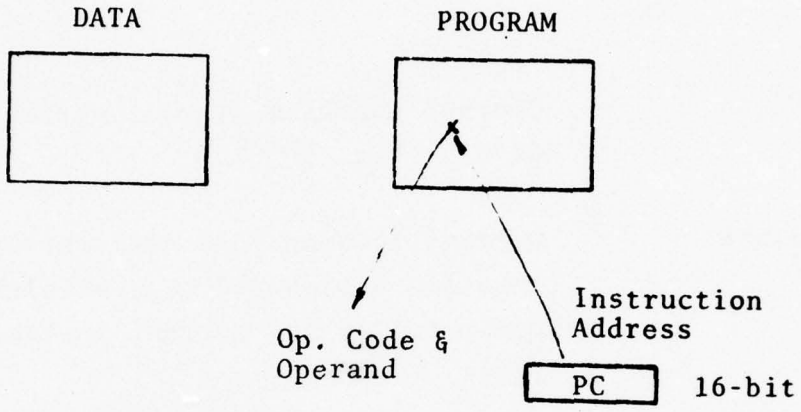


Figure A-1 Immediate (Single Fetch)

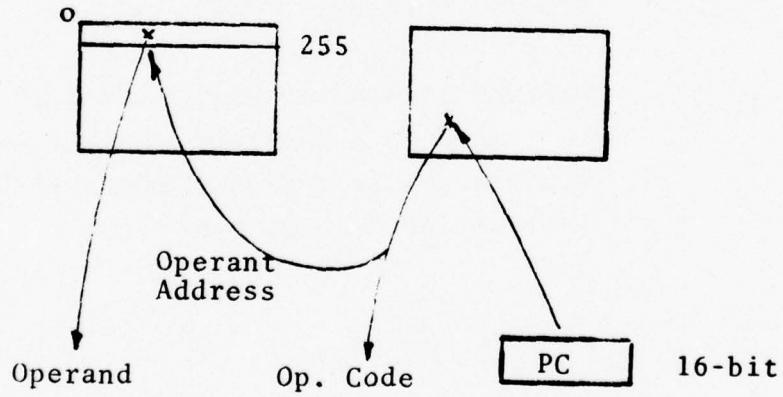


Figure A-2 Direct (Double Fetch)

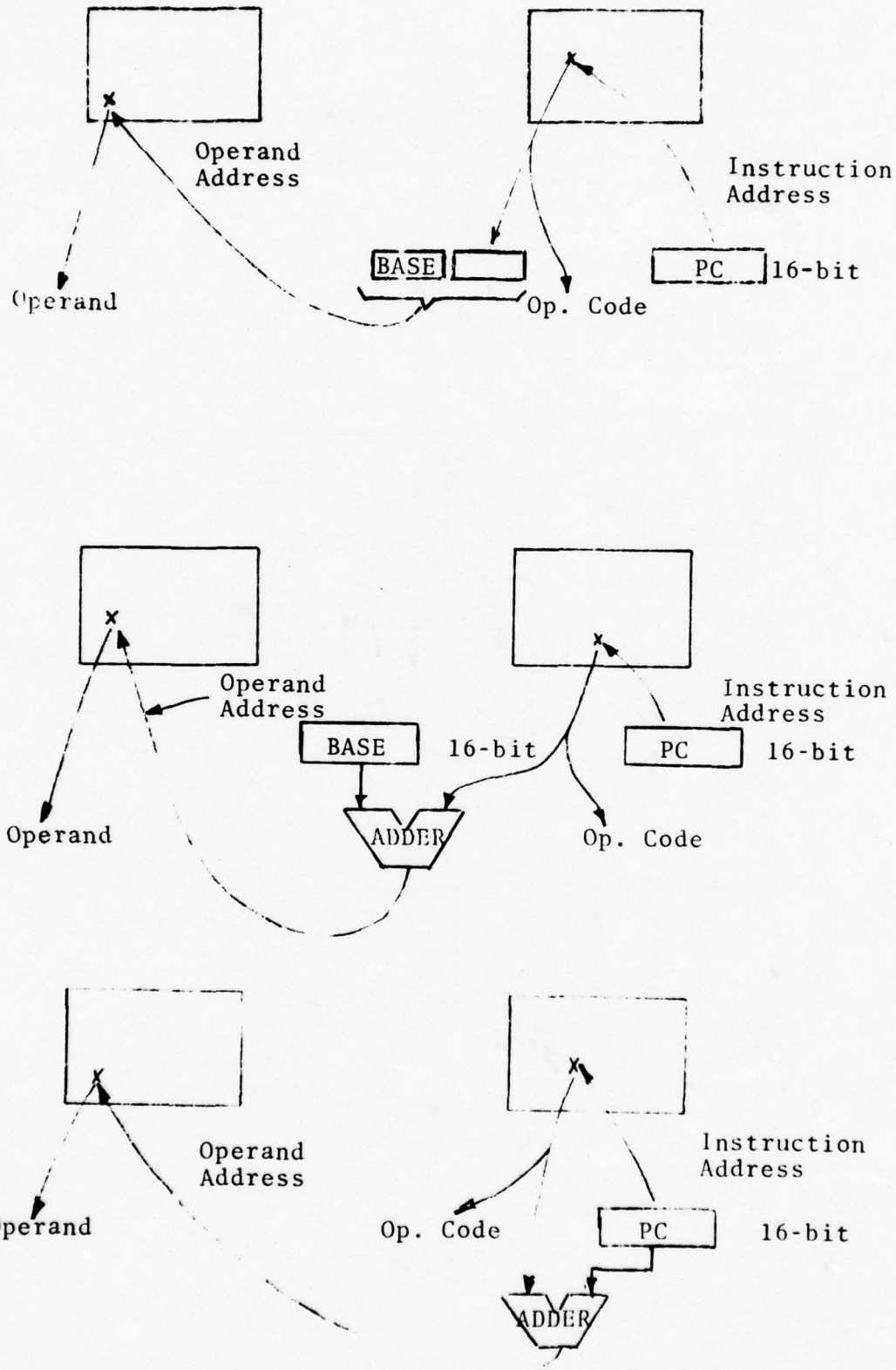
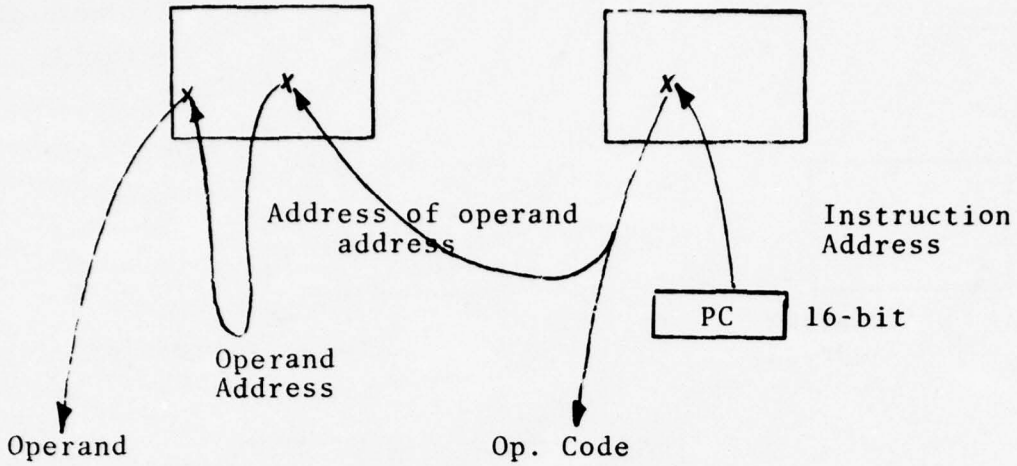
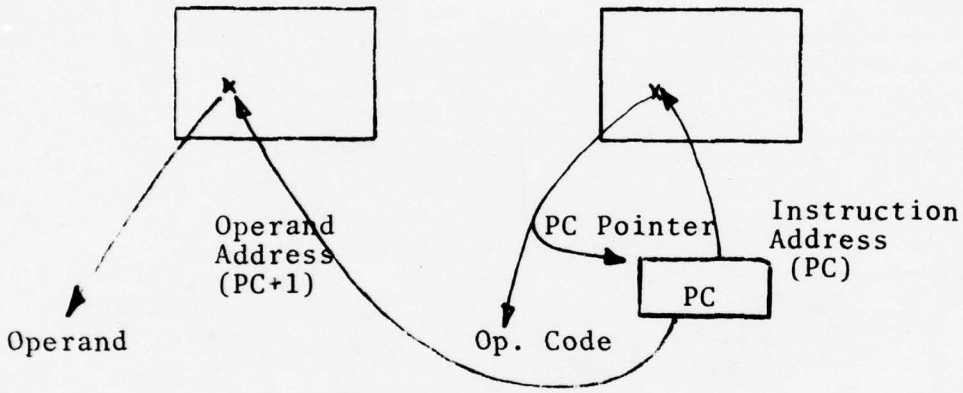


Figure A -3 Relative (Fetch, Add/Combine Fetch)





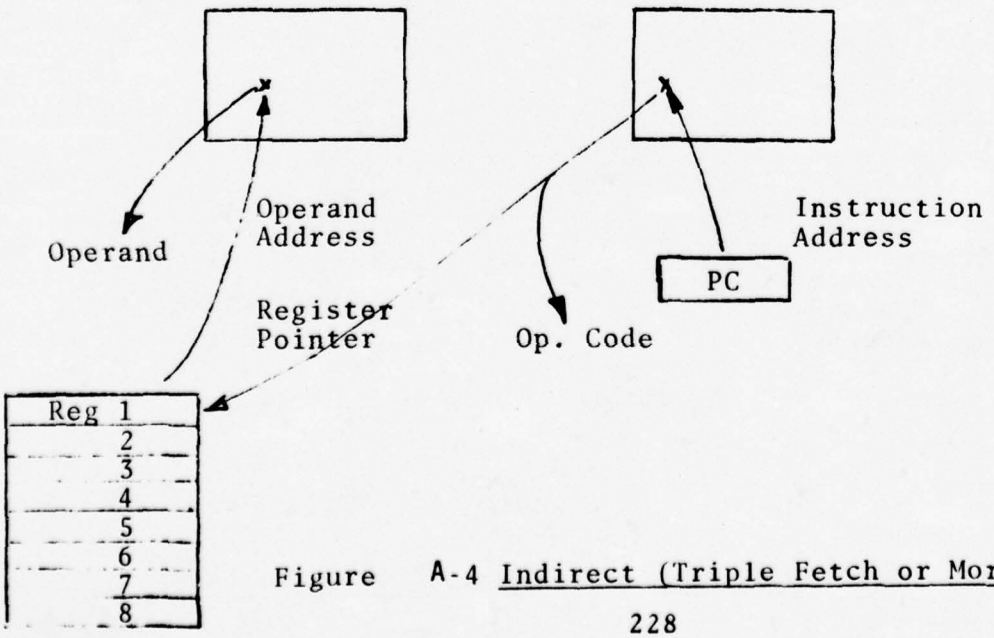


Figure A-4 Indirect (Triple Fetch or More for Multi-Level)

APPENDIX B

AMD 2901 - BASED μFFT - 1 μBUS MODULE

B1. INTRODUCTION

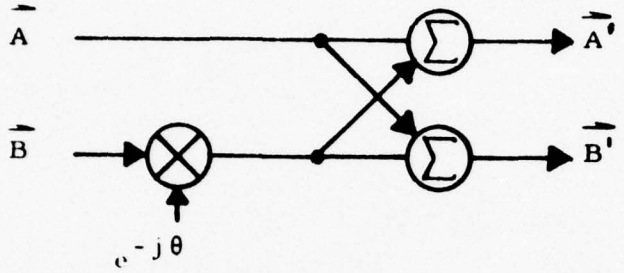
The well-known FFT algorithm efficiently carries out the Discrete Fourier Transform operation described in the Phase II Report, viz:

$$F(k) = \sum_{n=0}^{N-1} s(n)\, e^{-jnk\left(\frac{2\pi}{N}\right)} \qquad (1\text{-}1)$$

where	N	number of points
	s (n)	complex sample corresponding to nth point
	k	filter number
	F (k)	complex spectrum amplitude of kth filter

The FFT process by which the above equation is partitioned to minimize the necessary computations is described in Ref. R-24.

A complete FFT was shown to require $\frac{N}{2} \log_2 N$ complex arithmetic operations. The basic element of the FFT is the fundamental "Butterfly" operation which performs the necessary complex multiplications and summations, as shown below.



$$\vec{A} = I_A + j\,Q_A \qquad (1\text{-}2)$$

$$\vec{B} = I_B + j\,Q_B$$

$$e^{-j\theta} \quad \cos\theta - j\sin\theta$$

I In-phase component

Q Quadrature component

$$\vec{A'} \quad \left[I_A + (I_B \cos\theta + Q_B \sin\theta) \right] + j \left[Q_A + (Q_B \cos\theta - I_B \sin\theta) \right]$$

$$\vec{B'} \quad \left[I_A - (I_B \cos\theta + Q_B \sin\theta) \right] + j \left[Q_A - (Q_B \cos\theta - I_B \sin\theta) \right]$$

Figure B-1. Butterfly Representation

Note from the above equations that the requirement is for 4 multiplication operations and 6 add/subtract operations. The four multiplication operations are:

$$I_B \times \cos\theta,\ Q_B \times \cos\theta,\ I_B \times \sin\theta,\ \text{and}\ Q_B \times \sin\theta \qquad (1\text{-}3)$$

The six add/subtract operations are:

$$\text{Partial Sum \#1} = I_B \cos\theta + Q_B \sin\theta \qquad (1\text{-}4)$$

$$\text{Partial Sum \#2} = Q_B \cos\theta - I_B \sin\theta$$

$$I_A + \#1,\quad I_A - \#1,\quad Q_A + \#2,\quad Q_A - \#2$$

Diagramatically, the above mathematical process is represented as:

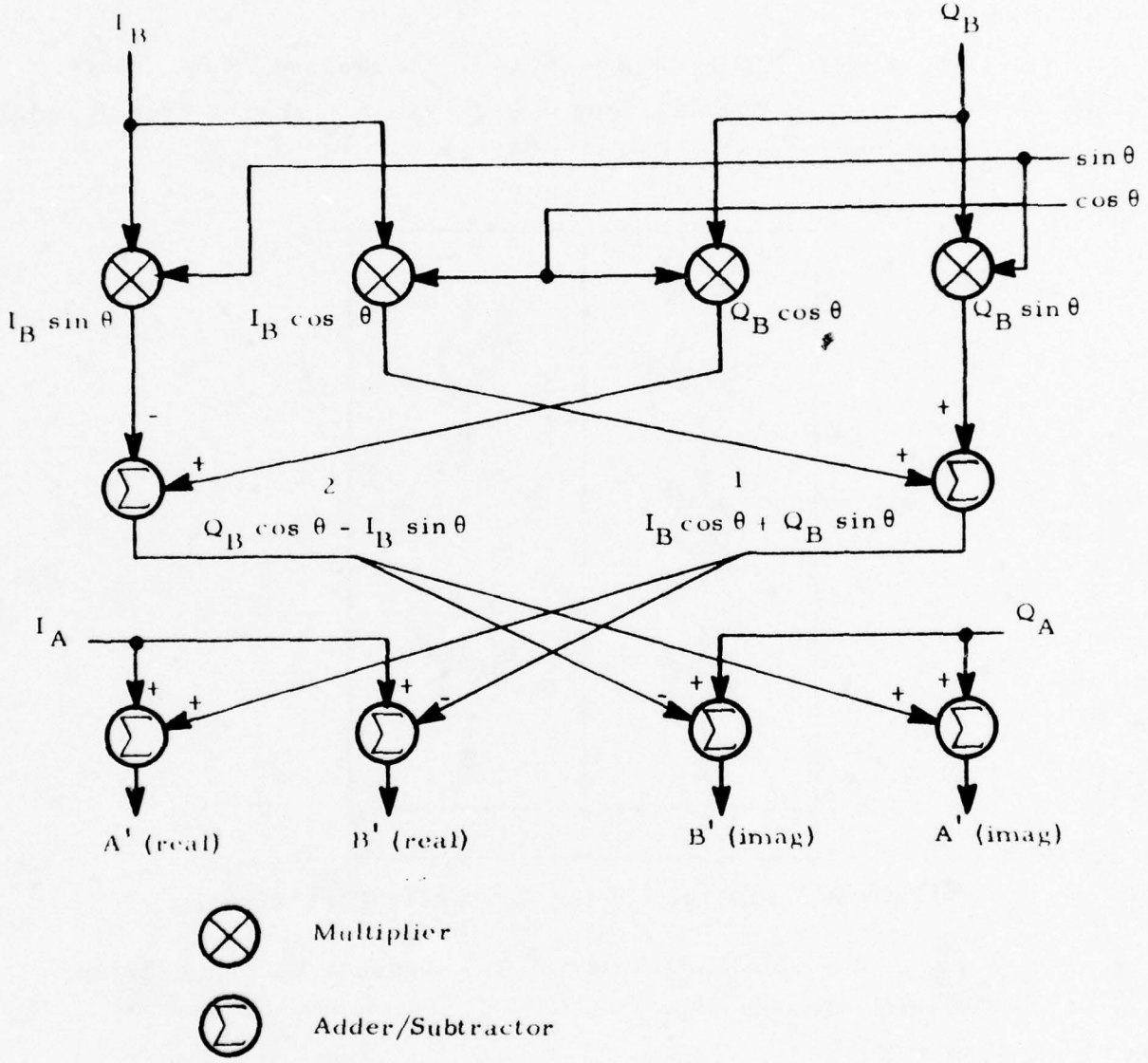


Figure B-2. Physical Representation of Butterfly

Examining the above diagram, we see that the Butterfly structure can be divided, in a hardware organization sense, in two ways--vertical slice or horizontal slice.

Two kinds of vertical slice organization are diagrammed below. There can be either two types of modules: Type 1 and Type 2 - or all Type 1 modules with one of the adders not used in half of these.

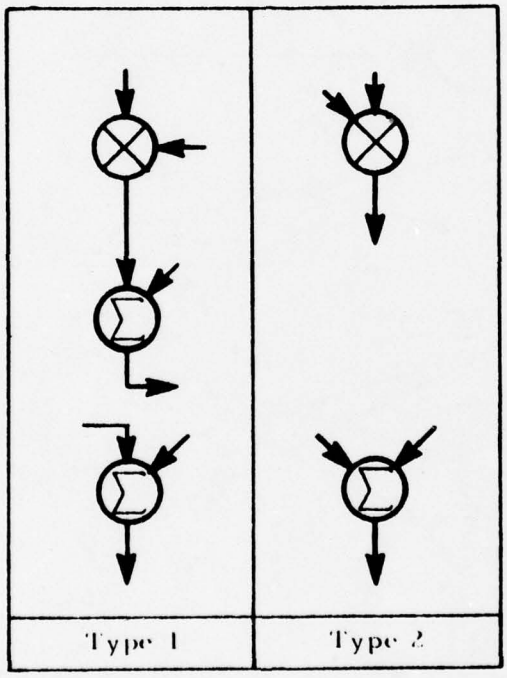


Figure B-3 Vertical Slice Butterfly Partitioning

There can be three kinds of horizontal slice modules as shown below. Or we can get along with only Type A and Type C, leaving out adders when a Type B is required.

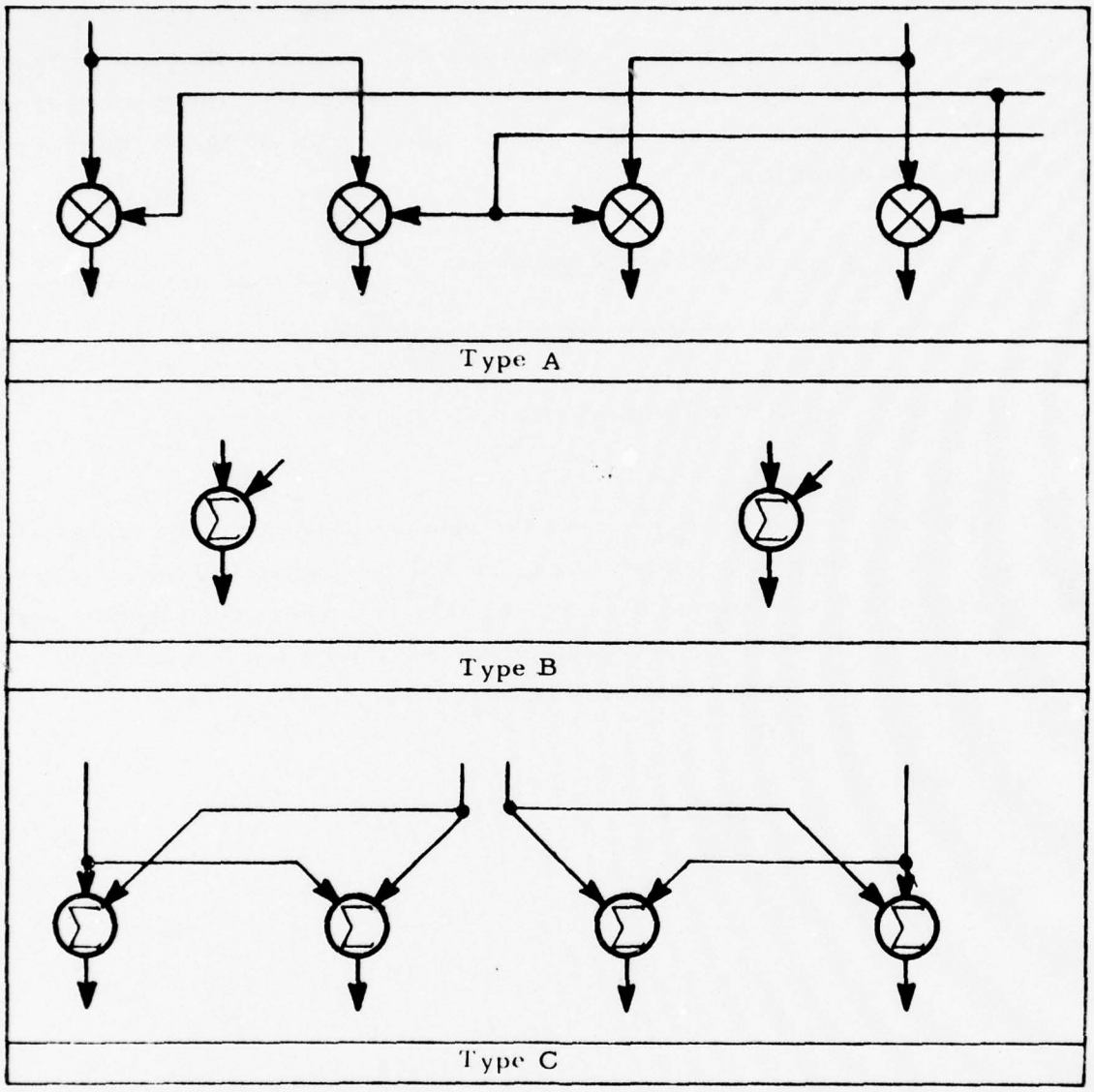


Figure B-4 Horizontal Slice Butterfly Partitioning

B.2 μ FFT MACROMODULE

The rationale for this module derives from the fact that in the FFT the I/O time is small compared to computation time. The μFFT macromodule approach also has this behavior. The processing time per range gate versus the number of points in the transform is given by the equation,

$$\text{processing time} \quad \frac{\text{number of points (n)}}{2} \times \log_2 n \times \text{time per Butterfly } (t_{BF})$$

The I/O time as a function of the number of points (n) is given by the equation,

$$\text{I/O time} \quad n \times 2 \times \text{time for I/O cycle } (t_{I/O}) \tag{2-1}$$

A plot of processing time versus the number of points in the transform with t_{BF} as a parameter is shown in Figure B-5. A plot of I/O time versus the transform size is shown in Figure B-6. The two plots used together can show the ratio of I/O time to processing time for the FFT. For a 4 μsec Butterfly time and a 60 ns cycle time of the memory:

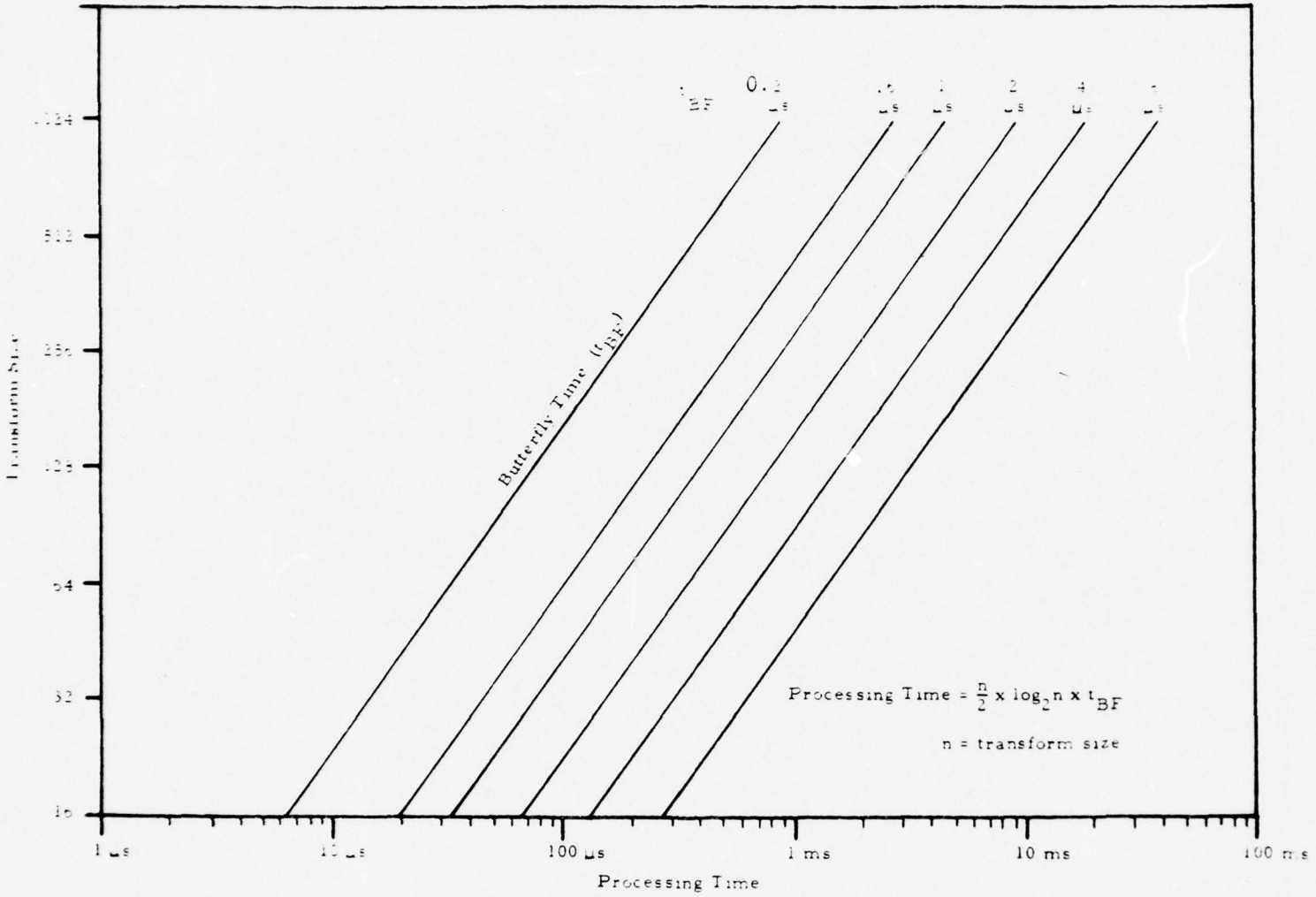


Figure B-5. Processing Time Versus Transform Size

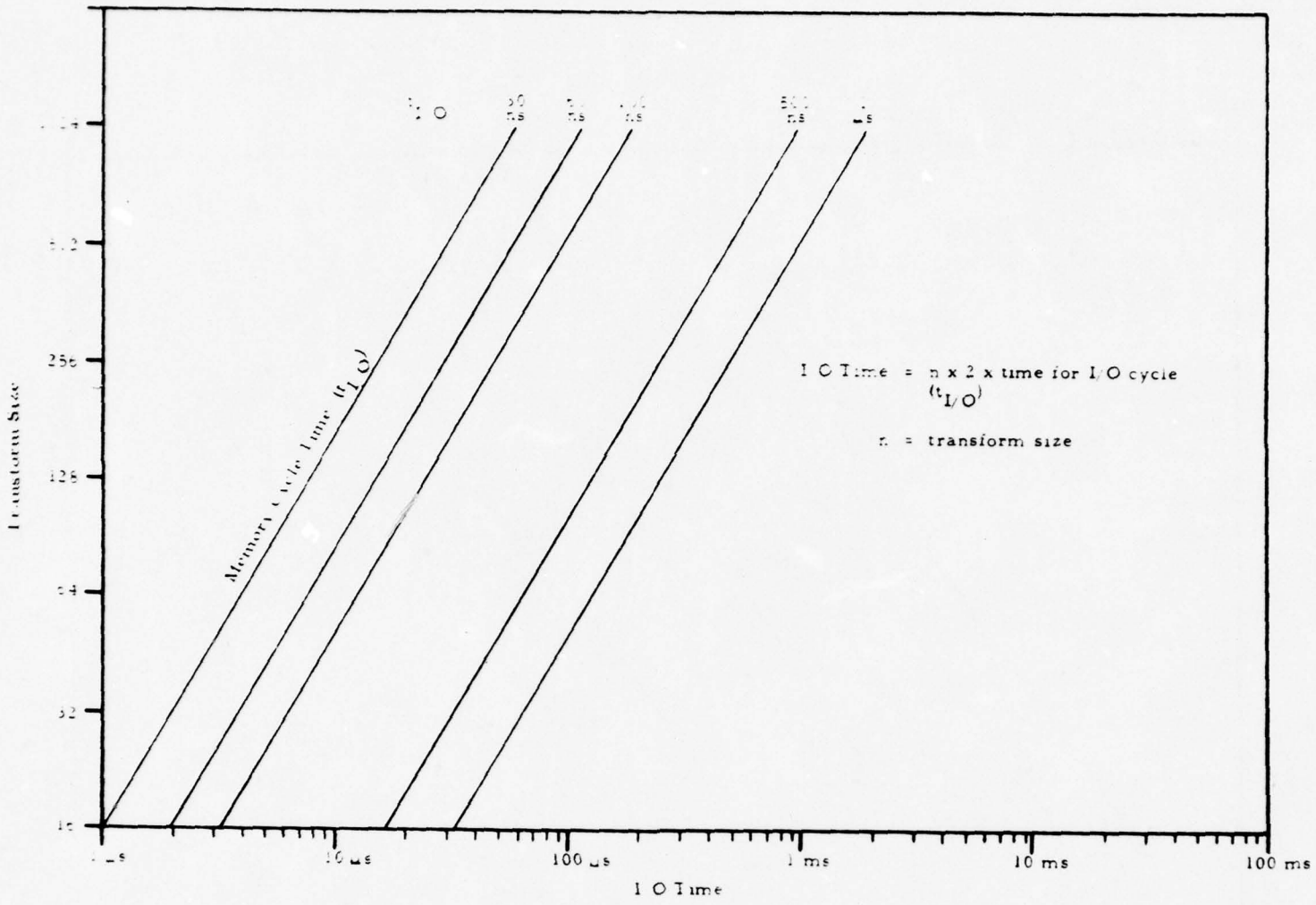


Figure B-6. I/O Time Versus Transform Size

TABLE B-1

Ratio of I/O Time to FFT Processing Time

(60 nsec RAM)

No. of pts	I/O Time (μ sec)	Processing Time (μ sec)	% = $\frac{\text{I/O Time}}{\text{Processing Time}}$
16	1.92	128	1.5
32	3.84	320	1.2
64	7.64	768	1.0
128	15.3	1792	0.85
256	30.72	4096	0.75
512	61.44	9216	0.67
1024	122.88	20480	0.6

B.3 2901-BASED μFFT MODULE

In the first paragraph, the FFT process was mathematically presented in equation 1-1 and further defined as requiring $\frac{N}{2} \log_2 N$ complex arithmetic operations. For a 64 point transform then $\frac{64}{2} \log_2 64$ or 192 complex arithmetic operations are required. This complex arithmetic operation is known as a butterfly as represented in Figure B-1. It was shown that the butterfly operation is composed of 4 multiply operations and 6 add/sub operations. The rationale for selecting the common element is based on the decomposing of the butterfly representation into these primative arithmetic operations of ADD and MULTIPLY.

Investigation into the programming of the basic 2 point transform kernel (Butterfly) of the FFT algorithm is not simply the 10 steps (i.e., 4 multiplies and 6 adds) but a function of the architecture, the organization of data in the memory (local store) and the address structure of the common element. Once the decision is made to construct the Butterfly to be composed of a single arithmetic logic unit (the 2901 microprocessor) and a hardware multiply element,

then it is necessary to consider efficiency of computation (Type 1 vertical organization omitting the second adder.) The micro instruction count for the basic 2 point transform kernel is in the order of 30 steps and is depicted in Table B-2. These 30 instructions assumed the use of a single microprocessor with a hardware multiply. The instruction count includes all the loads, stores, multiplies and adds that constitute the 2 point transform. Consider an execution time per instruction of 200 nanoseconds, then the time to process the butterfly is 30 x 200 ns = 6 μsec; for the 64 point transform which requires 192 x 6 μsec = 1.152 millisec. However, examination of the FFT reveals that the data could be subjected to a partitioned organization wherein pairs of values could be accessed simultaneously, thereby eliminating the usual instruction cycles needed for data access. This feature, in combination with the use of a multiplier element in a data path between the data memory and the summation microprocessor, results in a significant instruction saving. The realization of a technique to achieve this computational efficiency of 8 steps to perform the fundamental butterfly process is described below. (A butterfly cycle time of 8 x 200 ns or 1.6 μsec; 64 point transform executed in 192 x 1.6 μsec = 307.2 μsec; a computation efficiency of 30 steps down to 8 steps a reduction of 67%.

B.3.1 Parity Organized 2-point Transform Kernel (Butterfly)

The FFT calculation is a determination of N equally spaced transform components of a complex waveform from N equally spaced samples of that waveform. The equations which represent this calculation are shown in equation 1-1.

This section describes a 2-point FFT Butterfly processor (common element) with some advantages over the classical organization. Each pair of complex numbers is obtained in parallel from two single port random access memories (RAM). Vector rotations are always performed on the outputs from only one of the RAM's. The summation results are always returned to the same locations from which the original values were obtained. Addressing of the data RAM's (local store) and the rotation factor memory is relatively simple.

TABLE B-2

MICROPROGRAM FOR 2901-BASED BUTTERFLY USING HMPY MODULE

MICROINSTRUCTION			COUNT
LOAD	I_{32} in Latch	; Latch	1
MPY	Cos θ	; $I_{32}\cos\theta$ in Accum	1
STO	Temp 1	; Temp 1 = $I_{32}\cos\theta$	1
MPY	Sin θ	; $I_{32}\cos\theta$ in Accum	1
STO	Temp 2	; Temp 2 = $I_{32}\sin\theta$	1
LOAD	Q_{32} in Latch	; Latch	1
MPY	Cos θ	; $Q_{32}\cos\theta$ in Accum	1
STO	Temp 3	; Temp 3 = $Q_{32}\cos\theta$	1
MPY	Sin θ	; $Q_{32}\sin\theta$ in Accum	1
ADD	Temp 1	; Accum = $I_{32}\cos\theta + Q_{32}\sin\theta$	1
STO	Temp 1	; Temp 1 = $I_{32}\cos\theta + Q_{32}\sin\theta$	1
LOAD	Temp 3	; Latch	1
SUB	Temp 2	; Accum = $Q_{32}\cos\theta - I_{32}\sin\theta$	1
STO	Temp 2	; Temp 2 = $Q_{32}\cos\theta - I_{32}\sin\theta$	1
LOAD	I_0	; Latch	1
ADD	Temp 1	; $I'_0 = I_0 + (I_{32}\cos\theta + Q_{32}\sin\theta)$	1
STO	I_0	; Store in original location	1
SUB	Temp 1	; I_0 now in Accum	1
SUB	Temp 1	; Accum = $I_0 - (I_{32}\cos\theta + Q_{32}\sin\theta)$	1
STO	I_{32}	; Store in original location now I'_{32}	1
LOAD	Q_0	; Q_0 in Latch	1
ADD	Temp 2	; Accum = $Q_0 + (Q_{32}\cos\theta - I_{32}\sin\theta)$	1
STO	Q_0	; Store in original location now Q'_0	1
SUB	Temp 2	; Q_0 in Latch	1
SUB	Temp 2	; Accum = $Q_0 - (Q_{32}\cos\theta - I_{32}\sin\theta)$	1
STO	Q_{32}	; Store in original location now Q'_{32}	1
Index	I_0		1
Index	Q_0		1
Index	I_{32}		1
Index	Q_{32}		1
		TOTAL	30

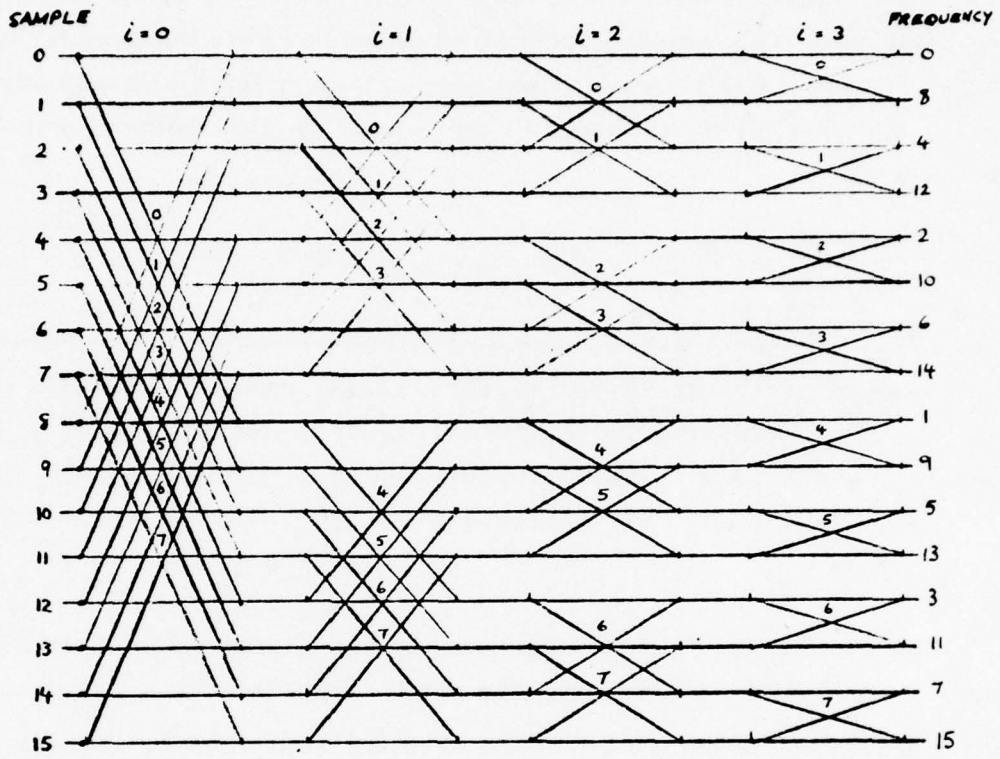
In performing the 2 point butterfly algorithm for an FFT the values of complex vectors are accessed from storage in a RAM module. One vector is rotated by multiplication with a complex unit vector. The rotated vector is then added to and subtracted from the unrotated vector, and the two resultant vectors are returned to RAM.

Each pass of the data through the butterfly processor requires a different order in accessing the data from the memory. If the classical order of data selection is used, and the storage is composed of single port random access memories (RAM), each of the two vectors must be accessed in series and temporarily stored in a "scratch pad" memory for use by the processor. The instruction cycles used to access the data constitute unwanted overhead.

For simplicity and clarity, a 16 point FFT data flow diagram is presented as Figure B-7. The representation is for a normal order processing of the complex samples to the final output in bit reversed order for the filter components. The W terms, rotational elements, are omitted. The sequency of operation suggested by this figure is defined by the indices i and j; for the lst pass i = 0 and j = 0 is the first butterfly calculation, j = 1 the second, on to the j = 7 butterfly calculation. In pass 2, i = 1, the j butterfly index proceeds from 0 through 7, and so on till all 32 butterfly calculations are accomplished to perform the 16 point FFT process. This operational sequence is shown below.

NORMAL SEQUENCE

j^{th} Butterfly		0	1	2	3	4	5	6	7
i^{th} pass	0	(0,8)	(1,9)	(2,10)	(3,11)	(4,12)	(5,13)	(6,14)	(7,15)
	1	(0,4)	(1,5)	(2,6)	(3,7)	(8,12)	(9,13)	(10,14)	(11,15)
	2	(0,2)	(1,3)	(4,6)	(5,7)	(8,10)	(9,11)	(12,14)	(13,15)
	3	(0,1)	(2,3)	(4,5)	(6,7)	(8,9)	(10,11)	(12,13)	(14,15)



NOTES:

16-POINT FFT - W TERMS OMITTED, INPUT IN NORMAL ORDER, OUTPUT BIT REVERSED.

i NUMBERS REPRESENT PASSES ($\text{Log}_2 N$ FOR 16 POINT = 0, 1, 2, 3)

NUMBERS THUS $\overset{2}{\times}$ REPRESENT jTH BUTTERFLY INDEX OF iTH PASS.

Figure B-7 16-Point FFT

The FFT algorithm does not order (j Butterfly operation) the sequences of calculation of the complex arithmetic operation within the pass (i^{th} pass) hence it is possible to change the order in the sequency and take advantage of the fact that for every Butterfly operation, the two indices involved differ in their parity.

PARITY SEQUENCE

j^{th} Butterfly		0	1	2	3	4	5	6	7
i^{th} pass	0	(0, 8)	(1, 9)	(2, 10)	(3, 11)	(4, 12)	(5, 13)	(6, 14)	(7, 15)
	1	(0, 4)	(9, 13)	(10, 14)	(3, 7)	(8, 12)	(1, 5)	(2, 6)	(11, 15)
	2	(0, 2)	(9, 11)	(8, 10)	(1, 3)	(12, 14)	(5, 7)	(4, 6)	(13, 15)
	3	(0, 1)	(8, 9)	(10, 11)	(2, 3)	(12, 13)	(4, 5)	(6, 7)	(14, 15)

This parity sequence above makes possible an organization of an n-point memory in 2 banks according to the parity of the indices.

Using the 16 point FFT as the example to explain the process, the memory organization of data will be described. The same subscripts (i, j) will be used for conformity throughout the explanation. The input is a complex pair of data samples reflecting the I and Q output from the radar receiver. The time sample indices are represented as a binary value for the index and is depicted as

S_0	S_1	S_2	S_3	---	S_7	S_8	---	S_{12}	---	S_{15}
0000	0001	0010	0011	---	0111	1000	---	1100	---	1111

The parity of the index is calculated which determines whether it is to be stored in an even parity local store or an odd parity local store. (Sample S_0 is even parity, Sample S_1 is odd parity, Sample S_2 is odd parity, --- , sample S_7 is odd parity, ---, sample S_{15} is even parity.) The 3 least significant bits are the indices (address) of the location in the even or odd parity memory where the data is initially stored. For pass i = 0 the complex samples are stored in the local store as shown.

Index	Even Parity Local Store	Odd Parity Local Store
0	S_0	S_8
1	S_9	S_1
2	S_{10}	S_2
3	S_3	S_{11}
4	S_{12}	S_4
5	S_5	S_{13}
6	S_6	S_{14}
7	S_{15}	S_7

Storage of Complex Samples for Pass i = 0.

The output pair to the arithmetic process to perform the complex operation (Butterfly) with its sequence is represented for pass i = 0 as (S_0,S_8), (S_9,S_1), (S_{10},S_2), (S_3,S_{11}), (S_{12},S_4), (S_5,S_{13}), (S_6,S_{14}), (S_{15},S_7) with storage back to the same location. The sequence for pass i = 1 is derived by inversion of the most significant bit of the index and using the contents of the location of that value for the sample of interest (odd parity local store). For example:

Index	Complement MSB	Sample of Odd Parity Local Store
000	100	Contents of Location 4 = S_4
001	101	Contents of Location 5 = S_{13}
010	110	Contents of Location 6 = S_{14}
011	111	Contents of Location 7 = S_7
100	000	Contents of Location 0 = S_8
101	001	Contents of Location 1 = S_1
110	010	Contents of Location 2 = S_2
111	011	Contents of Location 3 = S_{11}

This is depicted below, the results are returned to the same location as the output to the arithmetic process. Note that the even parity local store locations are not changed.

Index	Even Parity Local Store	Odd Parity Local Store
0	S_0	S_4
1	S_9	S_{13}
2	S_{10}	S_{14}
3	S_3	S_7
4	S_{12}	S_8
5	S_5	S_1
6	S_6	S_2
7	S_{15}	S_{11}

Memory Outputs to Arithmetic Process For Pass i = 1.

The output pair to the arithmetic process for pass i = 1 is (S_0, S_4), (S_9, S_{13}), (S_{10}, S_{14}), (S_3, S_7), (S_{12}, S_8), (S_5, S_1), (S_6, S_2), (S_{15}, S_{11}) with storage back to the same location.

The sequence for pass i = 2 is derived by complementing the next to the most significant bit of the index and using the contents of the location of that value for the sample of interest (from the odd parity local store only).

Index	Complement Next to MSB	Sample of Odd Parity Local Store
000	010	Contents of Location 2 = S_2 (original storage)
001	011	Contents of Location 3 = S_{11}
010	000	Contents of Location 0 = S_8
011	001	Contents of Location 1 = S_1
100	110	Contents of Location 6 = S_{14}
101	111	Contents of Location 7 = S_7
110	100	Contents of Location 4 = S_4
111	101	Contents of Location 5 = S_{13}

Index	Even Parity Local Store	Odd Parity Local Store
0	S_0	S_2
1	S_9	S_{11}
2	S_{10}	S_8
3	S_3	S_1
4	S_{12}	S_{14}
5	S_5	S_7
6	S_6	S_4
7	S_{15}	S_{13}

Memory Outputs to Arithmetic Process For Pass i = 2.

The output pair to the arithmetic process for pass i = 2 is (S_0, S_2), (S_9, S_{11}), (S_{10}, S_8) (S_3, S_1), (S_{12}, S_{14}), (S_5, S_7), (S_6, S_4), (S_{15}, S_{13}) with storage back to the same location.

The sequence for pass i = 3 is derived by complementing the next bit in the chain (for the 16 point transform, this is the LSB) addressing the contents of the location of that value for the sample of interest (from the odd parity local store only).

Index	Complement LSB	Sample of Odd Parity Local Store
000	001	Contents of Location 1 = S_1
001	000	Contents of Location 0 = S_8
010	011	Contents of Location 3 = S_{11}
011	010	Contents of Location 2 = S_2
100	101	Contents of Location 5 = S_{13}
101	100	Contents of Location 4 = S_4
110	111	Contents of Location 7 = S_7
111	110	Contents of Location 6 = S_{14}

Index	Even Parity Local Store	Odd Parity Local Store
0	S_0	S_1
1	S_9	S_8
2	S_{10}	S_{11}
3	S_3	S_2
4	S_{12}	S_{13}
5	S_5	S_4
6	S_6	S_7
7	S_{15}	S_{14}

Memory Outputs to Arithmetic Process For Pass i = 3.

The output pair to the arithmetic process for i = 3 is (S_0, S_1), (S_9, S_8), (S_{10}, S_{11}), (S_3, S_2), (S_{12}, S_{13}), (S_5, S_4), (S_6, S_7), (S_{15}, S_{14}) with storage back to the same location. At the completion of this pass, the frequency elements are stored in both memories in bit reversed order as depicted below (xxx bit reverse index value).

Address	Contents	Comment
000 0	f_0	Filter f_0 in location 0 of even parity memory
000 1	f_8	Filter f_8 in location 0 of odd parity memory
001 0	f_4	Filter f_4 in location 4 of odd parity memory
001 1	f_{12}	Filter f_{12} in location 4 of even parity memory
010 0	f_2	Filter f_2 in location 2 of odd parity memory
010 1	f_{10}	Filter f_{10} in location 2 of even parity memory
011 0	f_6	Filter f_6 in location 6 of even parity memory
011 1	f_{14}	Filter f_{14} in location 6 of odd parity memory
100 0	f_1	Filter f_1 in location 1 of odd parity memory
100 1	f_9	Filter f_9 in location 1 of even parity memory
101 0	f_5	Filter f_5 in location 5 of even parity memory
101 1	f_{13}	Filter f_{13} in location 5 of odd parity memory
110 0	f_3	Filter f_3 in location 3 of even parity memory
110 1	f_{11}	Filter f_{11} in location 3 of odd parity memory
111 0	f_7	Filter f_7 in location 7 of odd parity memory
111 1	f_{15}	Filter f_{15} in location 15 of even parity memory

Figure B-8 is a block diagram illustrating a method for calculating the addresses of the even parity memory, odd parity memory and the coefficient memory during an FFT conversion. Each address selected in each memory identifies two location

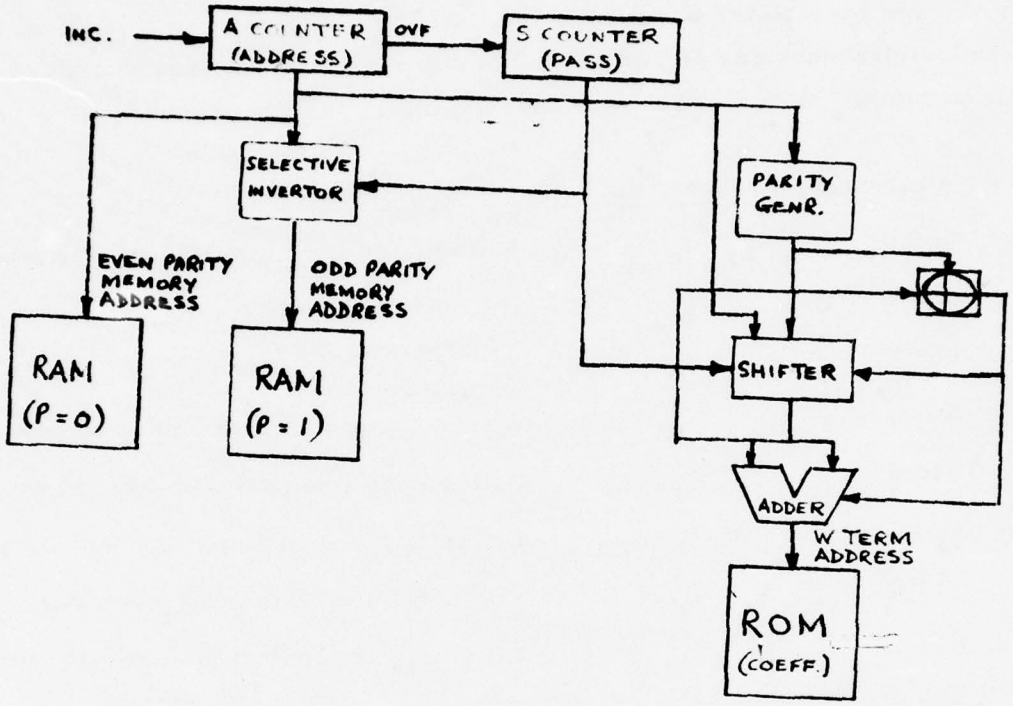


Figure B-8 μFFT MEMORY CONTROL BLOCK DIAGRAM

one containing the real value (I) of the complex vector selected and the other containing the imaginary value (Q) of the complex vector. Selection between the two locations is controlled by a separate ALU microprogram which is repeated for each butterfly calculation.

Referring to Figure B-8, the A counter provides the address A directly to the even parity memory. At the completion of the Butterfly calculation, the A count is incremented in preparation in the next butterfly calculation. The S counter is incremented at each overflow of the A counter. The S value (pass no.) is used to control the operation of the selective inverter and shifter. The selective inverter functions to invert one bit of the A value as directed by the S value and provides the address for the odd parity memory address. The parity generator, the shifter and adder are used to calculate the address of the coefficient memory under microprogram control.

Figure B-9 is a block diagram illustrating the μFFT-1 module for the signal processing function with the capability of performing the parity organized FFT conversion. The memory control is contained in the memory controller whose function is to:

1. Control the storage of data received from the micro bus in the odd parity and even parity memories (local store data memories).
2. Selection of data from the two data memories and the coefficient memory for use by the microprocessor.
3. The placement of the microprocessor results back into the two local stores, and
4. The delivery of data from the two data memories to the micro bus.

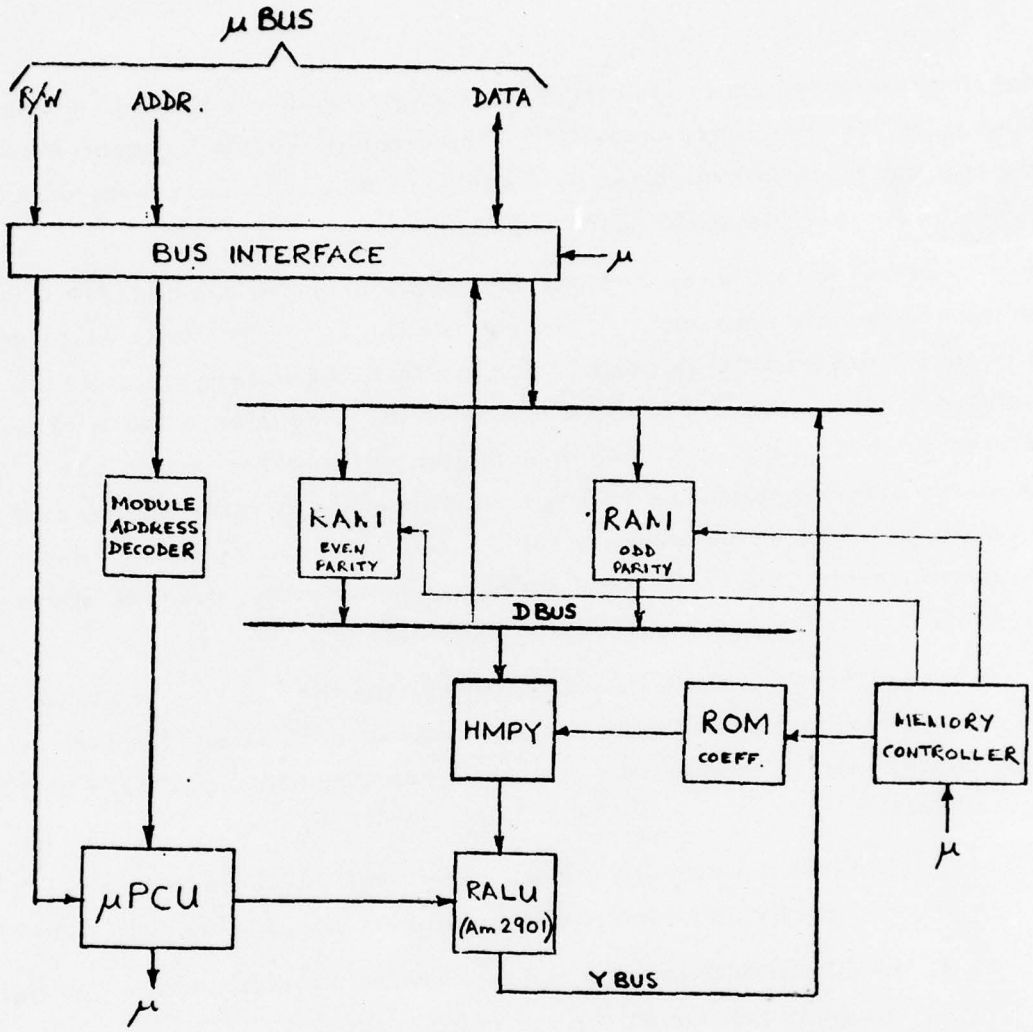


Figure B-9. BASELINE DESIGN OF 2901-BASED μFFT-1 MACROMODULE

(PARITY-ORGANIZED FFT)

The internal architecture of the 2901 microprocessor is well documented in the literature and its operation is described therein.

The task of performing the operation of the Butterfly is described in primative operations starting with the assumption: Stored in the even parity memory are the vector components for $\vec{A}$ $(I_A + jQ_A)$; stored in the odd parity memory are the vector components for $\vec{B}$ $(I_B + jQ_B)$; stored in the coefficient memory are the rotation terms, $\cos\theta$ and $\sin\theta$. These are represented as pictorially by

Even Parity Local Store

R_{EP}	J_{EP}
I_A	Q_A

Odd Parity Local Store

R_{OP}	J_{OP}
I_B	Q_B

Coefficient Memory

R_W	J_W
$\cos\theta$	$\sin\theta$

To perform the butterfly operation then the following steps are used (rotating the vector in the odd parity memory).

1. Perform $R_{OP} \times R_W \rightarrow$ Accum1

   Accum1 $= I_B \times \cos\theta$

2. Subtract $J_{OP} \times J_W$ from Accum1

   Accum1 $- Q_B \sin\theta \rightarrow$ Accum1

After this operation

   Accum1 $= I_B \cos\theta - Q_B \sin\theta$

3. Perform $J_{OP} \times R_W \rightarrow$ Accum2

   Accum2 $= Q_B \cos\theta$

4. Add $R_{OP} \times J_W$ to Accum2

$$\text{Accum2} + I_B \sin\theta \rightarrow \text{Accum2}$$

After this operation

$$\text{Accum2} = Q_B \cos\theta + I_B \sin\theta$$

5. Subtract Accum1 from R_{EP} and store in R_{OP} for R_{OP}'

$$I_A - (I_B \cos\theta - Q_B \sin\theta) \rightarrow R_{OP}$$

6. Add Accum1 to R_{EP} and store in R_{EP} for R_{EP}'

$$I_A + (I_B \cos\theta - Q_B \sin\theta) \rightarrow R_{EP}$$

7. Subtract Accum2 from J_{EP} and store in J_{OP} for J_{OP}'

$$Q_A - (Q_B \cos\theta + I_B \sin\theta) \rightarrow J_{OP}$$

8. Add Accum2 to J_{EP} and store in J_{EP} for J_{EP}'

$$Q_A + (Q_B \cos\theta + I_B \sin\theta) \rightarrow J_{EP}$$

Thus it can readily be seen that the entire butterfly operation can be programmed to perform the complex task in 8 steps. During the eighth instruction a status indication is sent from the microprocessor control to increment the next selection of the data and coefficient memory location.

B.3.2 Microprogram Word Structure

Details of the microinstruction control field are depicted in Figure B-10. The CE microprogram word is composed of instruction fields which are classified as either microprocessor control fields, memory control fields or instruction fields.

The microprocessor control fields designate the operation of the internal components of the common element's bit slice microprocessor configuration in implementing the parity organized FFT algorithm. These internal components include an arithmetic logic circuit (ALU), an accumulator (Q register), a save file (16 word 2-port RAM), an ALU source operand selector, and an ALU result destination control. (Internal elements of the 2900 microprocessor.)

The memory control fields control the access and placement of data in local memories, as well as the transfer of data to and from the processor system data bus. The local memories consist of the odd parity and even parity memories (data memories), the W coefficient memory (used when performing multiplications), and the macroinstruction memory (accessed by the memory control fields when programming the common element).

The instruction control fields indicate the location of the next microprogram word to be selected from the microprogram memory. Program branching in response to the microprocessor results is enabled by these instruction fields.

3.2.1 Microprocessor Control Fields

The microprocessor control fields are identified as: a) the source control field, b) the function control field, c) the destination control field, d) the file control fields, e) the accumulate carry field, and f) the multiplier control field.

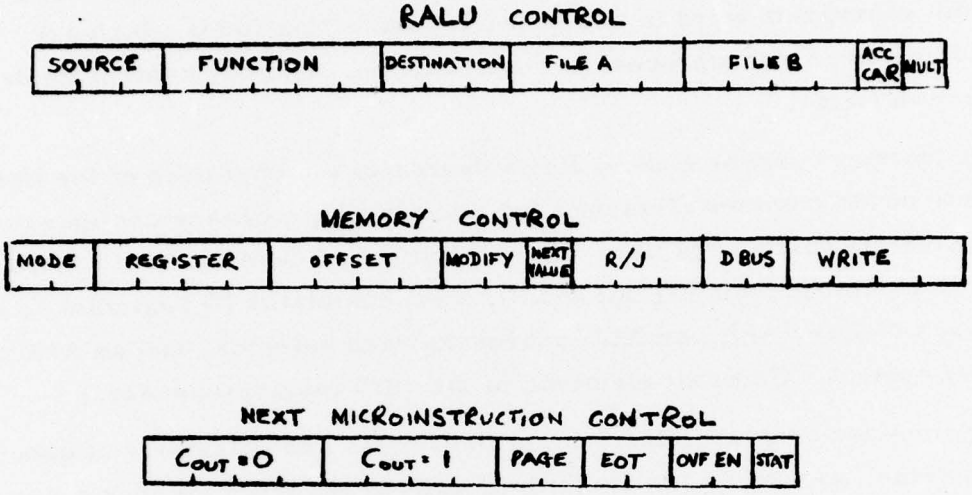


Figure B-10 Microprogram Word Field Allocation

a) Source Control Field (of the 2901 Microprocessor)

The source control field selects the sources of the R and S operands of the ALU from among the A and B output ports of the save file, the Q register, the microprocessor data input (D), or a value of zero (0). A tabulation of the eight selectable combinations follows:

Octal Code	R Selection	S Selection
0	A	Q
1	A	B
2	0	Q
3	0	B
4	0	A
5	D	A
6	D	Q
7	D	0

b) Function Control Field (Arithmetic Function Field of the 2901 Microprocessor)

Part of the function control field selects one of eight ALU functions to be performed on the designated R and S operands as listed as:

Octal Code	Function
0	S PLUS R
1	S PLUS $\overline{R}$ (≡ S MINUS R MINUS 1)
2	$\overline{S}$ PLUS R (≡ R MINUS S MINUS 1)
3	S OR R
4	S OR R
5	S AND $\overline{R}$
6	S EX-OR R
7	S EX-OR $\overline{R}$

A fourth bit in the function control field controls the carry input to the least significant bit (LSB) position during arithmetic functions (0, 1, or 2).

A fifth bit in the function control field is used to modify the selected ALU function based on the sign of the D value. If function 0, 1, 4, 5, 6, or 7 is selected, the R operand and the LSB carry input will be (1's) complemented when the value of D is negative. The intended use of this modification is to perform addition or subtraction of the absolute value of D.

c) Destination Control Field (of the 2901 Microprocessor)

The destination control field selects the destination of the ALU results (F) or its one-bit-position shifted value. The destination selection is either the Q register, the B input port of the save file, or the microprocessor data output (Y). One or more transfer paths may be selected as tabulated:

Octal Code	Transfer Paths
0	F to Q and F to Y
1	F to Y
2	F to B and A to Y
3	F to B and F to Y
4	1/2F to B and F to Y and 1/2Q to Q
5	1/2F to B and F to Y
6	2F to B and F to Y and 2Q to Q
7	2F to B

d) File Control Fields (Source, Source/Destination Addresses of the 2901 Microprocessor)

There are two file control fields, the A field and the B field. Each can concurrently select data stored in any of 16 memory locations of the save file output at its corresponding output port. The location at which data is entered into the save file (B input port) is controlled by the B file control field.

e) Accumulate Carry Field

The accumulate carry field functions to impress function 0 (R plus S) on the higher-order bit positions (those representing 2^{16} or higher) of the ALU and to shift the results at these bit positions one bit position to the left when they are transferred to the save file or Q register. The intended use of this operation is to enable the implementation of a division algorithm or a cordic vector angle determination algorithm.

f) Multiplier Control Field

The multiplier control field controls the source of the data impressed on the microprocessor data input (D). The two selections are either the D BUS data times the W factor from the multiplier, or the D BUS data direct from the D BUS.

Memory Control Fields

The memory control fields are identified as: a) the addressing mode field, b) the address register select field, c) the offset select field, d) the address modify field, e) the next address value field, f) the R/J control field, g) the D BUS control field, and h) the write control field.

a) Addressing Mode Field

The addressing mode field selects one of four modes for addressing the local memories of the common element. These are: the FFT mode, the weighting mode, the macroprogram W-select mode, and the S-count select mode.

The FFT mode is used to implement the Parity-Organized FFT algorithm with the W factors accessed from the ROM (FFT factor) portion of the W memory.

The weighting mode is used to select locations in the odd parity memory and even parity memory while selecting corresponding locations of the RAM portion of the W memory.

The macroprogram W select mode is similar to the weighting mode with the exception that the W factors are selected from the W memory by a macroprogram-generated address.

The S-count select mode is used to access data directly in any of the local data memories. The memory being accessed is determined by the S-count value.

In the first three modes the selection between the even parity memory and odd parity memory for the access of n and m data values* is determined by the parity of the A-count value less its least significant bit (A'). If the parity of the A'-count is even, the n data is accessed in the even parity memory and the m data is accessed in the odd parity memory. If the parity of the A'-count is odd, the alternate relationship holds.

In the FFT mode the even parity memory is addressed directly by the A-count value, and the odd parity memory and ROM portion of the W memory are addressed by S-count modified A-count values.

In the weighting mode the locations in the three memories are directly addressed by the A-count value plus the least significant bit of the S-count value. The indication of n and m are interchanged when the least significant bit of the S-count value is a ONE. The interchange of the n and m indication also holds for the macroprogram W-select mode.

The location in the S-count selected memory during the S-count selected mode is determined directly by the A-count value with the selected data identified as n data (m and w indications are not used).

b), c) Address Register Select and Offset Select Fields

A total of 16 memory control locations can each be program assigned as either an address register or an offset. The address register select field is used to select one of the assigned address registers, and the offset select field is used to select one of the assigned offsets. This arrangement

*n data value is the unrotated data element of the Butterfly
m data value is the rotated data element of the Butterfly

allows the interruption of an addressing sequency by another addressing sequence at any count value. When returning to the interrupted sequence, the count value continues from the point of interruption.

d) Address Modify Field

The address modify field selects one of three combinations of the value in the selected register and the selected offset, or a macroprogram-generated value, as the memory addressing word. The lower order bits of the memory addressing word comprise the A-count value and the higher-order bits comprise the S-count value. The four selections are:

0) Address register value
1) Address register value plus offset
2) Address register value minus offset
3) Macroprogram generated value

The last selection (3) is used to program the offset values and initialize the address register values.

e) Next Address Value Field

The next address value field controls the value returned to the selected address register at the completion of the current instruction. The selection is between either no change in value or the modified value selected by the address modify field.

f) R/J Control Field

The R/J control field is intended for use when the data stored in the odd parity memory and even parity memory consist of complex values. In this case the real component (R) of each complex value is located in an even-addressed location of the memory, and the corresponding imaginary component (J) is located in the corresponding odd-addressed location of the memory.

Only even valued addressing words are used to address each pair of data (or factor) words which comprise a complex value in the even parity, odd parity, or W memories. Selection between the R and J components of each complex value is provided by the R/J control field. One bit separately controls each of the n, m, and W factor values selected from the memories.

When an odd-valued addressing word is used to select complex values, the indications of the n and m control bits of the R/J control field are complemented. Thus the option exists for selection between R and J components of the n and m data by either the addressing word, the R/J control field, or a combination of both. (When operating in the weighting addressing mode, with the R/J control field held constant the R/J selection made by the addressing word, the same weighting factor will be selected for both components of each complex value.)

g) D BUS Control Field

The D BUS control field selects the source of data impressed on the D BUS of the common element. The four selections are: n data, m data, Y BUS data (from the microprocessor output), or the DATA BUS.

The last selection is used when the common element is in the receiving operation mode. If the common element is not in the receiving operation mode this selection will access the memory addressing word instead of the DATA BUS.

h) Write Control Field

The write control field controls storage of data from the Y BUS. A separate control bit enables or disables storage in or transfer to, its indicated destination. The four destinations are: n data storage, m data storage, W factor storage (when the RAM portion of the W memory is selected), and the DATA BUS. Y BUS data is transferred to the DATA BUS when the common element is in a transmitting operation mode.

μInstruction Control Fields

The instruction control fields are identified as: a) the next instruction elds, b) the end of task field, c) the overflow enable field, and d) the atus mark field.

a) Next Instruction Fields

The location of the next microprogram instruction is determined by the condition of the carry out from the 2^{15} stage of the microprocessor's ALU. The next instruction fields indicate the location of the next instruction in the microprogram memory for each of the two carry conditions.

The two possible locations are always located within the same 16 word page of the memory. The C_{OUT} = 0 field indicates the word in the page for a carry value if ZERO and the C_{OUT} = 1 field indicates the word in the page for a carry value of ONE. If the selection of the next instruction is independent of the value of the carry the two fields will contain the same value.

The page field indicates one of four page locations of the indicated instructions. These are:

0) the current page
1) the next page
2) two pages back
3) one page back

In the infrequent case in which the number of instructions of a task microprogram is greater than two pages, a large displacement between the last and return instructions can be avoided by the location of the first half of the instructions in even numbered memory locations in ascending order and the location of the last half of the instructions in the odd-numbered memory locations in descending order.

b) End of Task Field

The end of task field selects the condition which identifies the present instruction as the last instruction to be performed in the present task. If the condition is present an EOT indication will be transferred to the macroprogram control to initiate the next task. If the condition is not present the present task will continue with the microinstruction indicated by the next instruction fields. The four conditions are:

0) not end of task
1) unqualified end of task
2) end of task if addressing word corresponds to a macroprogram-generated word
3) end of task if Y BUS data corresponds to a macroprogram-generated word

c) Overflow Enable Field

The overflow enable field contains two control bits. The enable control bit is used to enable or disable setting of the overflow register if a value overflow is produced by the lower 16 stages of the microprocessor's ALU. The reset control bit is used to reset the overflow register.

d) Status Mark Field

The status mark field is used to enable the transfer of an OVF indication to the macroprogram control, if the overflow register has been set. This indication is used by the macroprogram control to inititate the action required by the overflow condition.

DISTRIBUTION LIST

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217
- ONR 212 (CDR P. R. Hite) (6)
- ONR 230 (Mr. A. White) (1)
- ONR 102IP (6)
- ONR 221 (Mr. J. Trimble) (1)

Office of Naval Research Branch Office
495 Summer Street
Boston, Mass 02210 (1)

DCASR
Raytheon Company
Wayside Ave.
Burlington, Mass 01803 (1)

U.S. Naval Research Laboratory
Washington, D. C. 20375
- NRL 2627 (6)
- NRL 5216 (Mr. L. Palkuti) (1)
- NRL 5260 (Dr. D. Barbe) (1)

Defense Documentation Center
Bldg. 5, Cameron Station
Alexandria, VA 22314 (12)

Naval Air Systems Command
Washington, D. C. 20361
- NAIR-03P2 (Mr. J. W. Malloy) (1)
- NAIR-03P3 (Mr. R. H. Krida) (1)
- NAIR-350F (Mr. R. J. Wasneski) (1)
- NAIR-360B (Mr. B. A. Zempolich) (1)
- NAIR-360E (Mr. V. A. Tarulis) (1)
- NAIR-52022B1 (Mr. G. B. Cudd) (1)
- NAIR-533DX (Mr. R. S. Entner) (1)
- PMA-263 (CAPT W. M. Locke) (1)

Naval Air Development Center
Warminster, PA 18974
- NADC-2031 (Mr. C. M. Nowicki) (1)
- NADC-2072 (Mr. W. M. Norr) (1)

Naval Material Command
Washington, D. C. 20362
- NMAT-0321 (Mr. S. Jacobson) (1)
- NMAT-0321 (Mr. C. Lyons) (1)

Naval Sea Systems Command
Washington, D. C. 20362
- NSEA-03132 (Mr. R. Cuddy) (1)
- NSEA-0341 (Mr. T. Tasaka) (1)
- NSEA-393M2 (CDR J. Miltonberger) (1)
- PMS-406 (CAPT M. T. Greeley) (1)

Director of Defense Research and Engineering
The Pentagon
Washington, D. C. 20350
- Mr. M. Keller (Rm. 3E1081) (1)
- LTC G. Kopcsak (Rm. 3D1089) (1)

Naval Avionics Facility
Indianapolis, Ind 46218
- NAFI-072.9 (Mr. R. H. Huss) (1)
- NAFI-D072 (Mr. W. Bridges) (1)
- NAFI-D824 (Mr. C. W. Hagen) (1)
- NAFI-D824 (Mr. J. Carr) (1)
- NAFI-D900 (Mr. R. Barnett) (1)
- NAFI-D924 (Mr. G. Forsee) (1)

Naval Ocean Systems Center
San Diego, CA 92152
- NOSC-19 (Mr. S. Maynard) (1)
- NOSC-722 (Ms. N. S. Mathis) (1)
- NOSC-8222 (Mr. D. Wilcox) (1)
- NOSC-9102 (Mr. W. Dejka) (1)
- NOSC-923 (Mr. E. C. Urban) (1)
- NOSC-923 (Mr. J. F. Poulson) (1)
- NOSC-923 (Mr. E. S. Holland) (1)
- NOSC-9232 (Dr. R. Martinez) (1)
- NOSC-9232 (Mr. R. D. Peterson) (1)

Naval Surface Weapons Center
Dahlgren Laboratory
Dahlgren, VA 22448
- DF-34 (Mr. R. Holden) (1)
- DF-34 (Mr. R. Dorsey) (1)
- DK-21 (Mr. F. J. Stevens) (1)

Chief of Naval Operations
The Pentagon
Washington, D. C. 20350
- NOP-982E1 (CDR K. H. Cox) (1)
- NOP-987P3 (Mr. W. B. McCall) (1)

Naval Weapons Center
China Lake, CA 93555
- NWC-01 (Mr. R. M. Hillyer) (1)
- NWC-015 (Dr. R. Cartwright) (1)
- NWC-033 (Mr. C. Neal) (1)
- NWC-395 (Mr. W. G. Hueber) (1)
- NWC-3130 (Mr. L. Lakin) (1)
- NWC-3132 (Mr. R. Hawkins) (1)

Naval Electronic Systems Command
Washington, D. C. 20360
- ELEX-304 (Mr. J. Cauffman) (1)

Naval Surface Weapons Center
White Oak Laboratory
Silver Spring, MD 20910
- WA-32 (Mr. A. Kolodzinski) (1)

National Aeronautic and Space Administration
Headquarters
Washington, D. C. 20546
- NASA-KC (Mr. W. E. McInnis) (1)

National Aeronautic and Space Administration
Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasedena, CA 91103
- NASAJPL-114-122 (Mr. P. E. Lecoq) (1)
- NASAJPL-114-122 (Mr. M. Ebersole) (1)
- NASAJPL-158-205 (Mr. R. E. Covey) (1)
- NASAJPL-T1180 (Mr. R. W. Scott) (1)

Institute for Defense Analysis
400 Army Navy Drive
Washington, D.C. 22202
- Dr. T. C. Bartee (1)
- Mr. G. W. Preston (1)

Air Force Avionics Laboratory
Wright Patterson AFB, Ohio 45433
- AFAL/DHE (MAJ J. A. Decaire) (1)
- AFAL/DHM (Mr. R. E. Conklin) (1)

Armament Development and Test Center
Eglin AFB, Florida 32542
- ADTC/DL (COL A. D. Brown) (1)
- ADTC/SD7 (Mr. G. Kirby) (1)
- ADTC/ADDE (Dr. J. W. Jones) (1)
- ADTC/DLMI (LCOL Britton) (1)
- ADTC/DLMM (MAJ R. Haney) (1)
- ADTC/DLMT (MAJ E. Halprin) (1)

Air Force Systems Command
Andrews AFB, Md 20334
- AFSC/DLCA (Mr. P. Sandler) (1)
- AFSC/SDZ (COL R. Loukota) (1)
- AFSC/XRL (COL F. Smith) (1)

U.S. Army Electronics Command
Fort Monmouth, N. J. 07703
- NL-BP (Mr. D. Haratz) (1)

U. S. Army Missile R&D Command
Redstone Arsenal, Ala 35809
- DRDMI-TG (Mr. J. B. Huff) (1)
- DRDMI-TGG (Mr. D. Ciliax) (1)
- DRDMI-TE (Mr. D. Holter) (1)
- DRDMI-EA (Mr. C. Riley) (1)
- DRCPM-PE-X (Mr. W. Jann) (1)
- DRCPM-MD (Mr. E. Wood) (1)

Harry Diamond Laboratory
2800 Powder Mill Road
Aldephi, MD 20783
- Dr. Carter (1)

The Analytical Sciences Corporation
6 Jacob Way
Reading, Mass 01857
- Mr. C. F. Price (1)